Análisis y Diseño del Subsistema de Ejercicios para el Laboratorio Virtual de Sistemas Operativos

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE
INGENIERÍA EN CIENCIAS INFORMÁTICAS

Autor:

Yanet Santiago Pérez

Tutor:

Ing. Carlos Y. Hidalgo García

Ciudad de La Habana, Cuba

2011

Declaración de Autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yanet Santiago Pérez

_____

Firma del Autor

Carlos Y. Hidalgo García

_____

Firma del Tutor

Dedicatoria

En especial a mi abuela.

A mis padres, lo más lindo, valioso y maravilloso que tengo en la vida.

A mi familia en general.

Agradecimientos

A mi familia, gracias a ellos he llegado a ser lo que soy.

A Jaime, por su ayuda y comprensión en los momentos más difíciles.

Al tutor de este trabajo por todo su apoyo.

A mis amigos y compañeros de estudio.

A todos los que de una manera u otra han ayudado en mi formación personal y profesional.

Resumen

El desarrollo alcanzado en las Tecnologías de la Información y la Comunicación (TIC) ha posibilitado la creación de herramientas que aumentan la calidad del proceso de enseñanza – aprendizaje. Entre estos sistemas destacan los laboratorios virtuales como medios accesibles y adaptables para poner en práctica conocimientos.

En la Facultad 3 de la Universidad de las Ciencias Informáticas se lleva a cabo el proyecto Laboratorio Virtual de Sistemas Operativos con el objetivo de dotar a la asignatura de una herramienta de apoyo al aprendizaje colaborativo y semipresencial, para respaldar el nuevo modelo de formación enfocado al aprendizaje a través de la producción – investigación. Con el fin de facilitar la implementación de un sistema mediante el cual los estudiantes puedan consolidar los conocimientos en dicho laboratorio y a su vez verificar el progreso de su autoaprendizaje, además que los profesores puedan comprobar los niveles alcanzados por los mismos en vistas a corregir y controlar el proceso educativo, se realiza el análisis y diseño del Subsistema de Ejercicios que ofrecerá materiales de evaluación (ejercicios y cuestionarios) para satisfacer los requerimientos de la asignatura y contribuir al perfeccionamiento de la misma en función de aumentar la calidad del egresado UCI.

En el desarrollo de este trabajo se hace uso de la metodología RUP, el lenguaje de modelado UML y la herramienta CASE Visual Paradigm para la construcción de los modelos de análisis y diseño. Se utilizan además técnicas de la ingeniería de requisitos para la captura, especificación y validación de los mismos y algunos patrones de diseño y casos de uso para la creación del diagrama de casos de uso del sistema y el diagrama de clases de diseño.

Palabras claves: laboratorio virtual, evaluación, autoevaluación, análisis, diseño.

Índice

Índice de Figuras

Índice de Tablas

Introducción

El desarrollo alcanzado en las Tecnologías de la Información y la Comunicación (TIC), posibilita el diseño de herramientas para aumentar la calidad del proceso de enseñanza - aprendizaje y elevar la capacidad de autoaprendizaje y autoevaluación de los estudiantes.

Los Entornos Virtuales de Aprendizaje (EVA) por ejemplo, permiten la gestión de manera no presencial de dicho proceso, en el cual están inmersos tanto educandos como educadores. Por otra parte, los entornos simulados o laboratorios virtuales, facilitan el uso de prácticas y ejercicios como métodos de consolidación de conocimientos. Estas nuevas formas de enseñanza eliminan los obstáculos que impone la localización física de la información, y su flexibilidad permite que se adapte a las características concretas de sus usuarios.

La Universidad de las Ciencias Informáticas (UCI), creada como parte del proceso de informatización de la sociedad cubana, con el objetivo de formar profesionales en dicha rama e impulsar el desarrollo de software en el país, es uno de los centros pionero en el uso de nuevos modelos de formación enfocados al aprendizaje a través de la producción – investigación, estratégicamente diseñado para dar cumplimiento a sus tareas como centro productor. En este nuevo modelo cobra vital importancia el uso de las TICs como herramientas de autoaprendizaje, que permitan fortalecer el aprendizaje semipresencial y así disponer de un mayor tiempo para dedicar a producir e investigar.

Sin embargo, no todas las asignaturas que conforman el plan de estudios del ingeniero informático formado en la UCI cuentan con materiales didácticos suficientes que faciliten el desempeño de la semipresencialidad. Es el caso de la asignatura de Sistemas Operativos que no goza de buena interacción con las TICs y depende de numerosas actividades presenciales para entregarle al estudiante su fuerte contenido teórico, entrando en conflicto con el nuevo modelo de formación.

En virtud de satisfacer esta necesidad es concebido, en la Facultad 3 de la universidad, un Laboratorio Virtual de Sistemas Operativos que persigue brindarle a los estudiantes un espacio donde puedan gestionar los contenidos de la asignatura y mediante prácticas simuladas logren un entendimiento superior sobre los complejos elementos que conforman un sistema operativo. El proyecto está integrado por 17 módulos y 6 subsistemas. Sin embargo, ninguna de estas aplicaciones le permite al alumno poner en práctica los conocimientos adquiridos para comprobar el progreso de su aprendizaje y que a su vez el profesor verifique los niveles alcanzados por sus estudiantes en vistas a corregir y controlar el proceso educativo.

Surge entonces el siguiente **problema investigativo**: ¿Cómo modelar un subsistema que garantice la consolidación de habilidades y conocimientos en el Laboratorio Virtual de Sistemas Operativos, para su futura implementación?

Constituye **objeto de estudio** de esta investigación el proceso de desarrollo de software, restringiendo el **campo de acción** al flujo de trabajo Análisis y Diseño del proceso de desarrollo de software.

Como **objetivo general** se plantea obtener los modelos de análisis y diseño del subsistema de ejercicios para el Laboratorio Virtual de Sistema Operativos.

Para el cumplimiento del objetivo propuesto se trazan los siguientes **objetivos específicos**:

- Identificar las funcionalidades y cualidades que debe poseer el subsistema.
- Realizar los modelos de análisis y diseño.
- Validar la solución propuesta.

Se define como **idea a defender**: Con la obtención de los modelos de análisis y diseño del subsistema de Ejercicios del Laboratorio Virtual de Sistemas Operativos, se facilitará su posterior implementación.

El presente trabajo de diploma consta de tres capítulos y está estructurado de la siguiente manera:

El primer capítulo enmarca la **Fundamentación Teórica** de la investigación, donde se analizan los diferentes tipos de laboratorios virtuales y las ventajas y desventajas de su uso. Se hace un estudio de las técnicas de evaluación que se implementan con mayor frecuencia en los entornos virtuales e incluye la descripción del módulo de autoaprendizaje de la plataforma de tele-enseñanza AulaWeb, con el objetivo de valorar sus funcionalidades y obtener una aproximación a la solución que se desea. Para lograr una adecuada administración de los requisitos se profundiza en los principales aspectos de la disciplina Ingeniería de Requisitos, además se analizan las herramientas y tecnologías que serán usadas para dar solución al problema de esta investigación.

En el segundo capítulo, **Descripción de la solución**, son representadas las clases conceptuales del negocio en un modelo de dominio. Se especifican los requisitos funcionales y no funcionales y se modela el diagrama de casos de uso del sistema. Una vez precisadas las funcionalidades se construye el modelo de análisis para establecer la relación entre clases y objetos del dominio, especificando en el diseño los métodos que permiten automatizar cada funcionalidad del subsistema.

En el tercer capítulo, **Validación de la solución**, se evalúa la calidad de los artefactos generados como parte de la solución propuesta, utilizando técnicas de validación de la ingeniería de requisitos y métricas para medir la calidad del software.

Capítulo 1. Fundamentación Teórica

Introducción

El presente capítulo brinda una visión general de los aspectos relacionados con los laboratorios virtuales, sus características, ventajas e inconvenientes. Recoge además una panorámica de la técnica de evaluación que se implementa con mayor frecuencia en los entornos virtuales e incluye la descripción del módulo de autoaprendizaje de la plataforma de tele-enseñanza AulaWeb, con el objetivo de valorar sus funcionalidades y obtener una aproximación a la solución que se desea. Presenta las actividades y técnicas de la disciplina Ingeniería de Requisitos que se consideran para la obtención, documentación y validación de requisitos en este trabajo y las posibles herramientas, metodologías y lenguajes de modelado a utilizar en el desarrollo de la solución.

1.1 Laboratorio virtual

Son diversas las definiciones de *Laboratorio Virtual* que brindan varios conocedores del tema, entre las que se pueden encontrar:

“Conjunto de experimentos virtuales que tiene como objetivo preparar al usuario para obtener el máximo rendimiento de un laboratorio real.” [1]

"Simulaciones de prácticas manipulativas que pueden ser hechas por el estudiante lejos de la universidad y el docente." [2]

“El laboratorio virtual es un tipo de colaboración centrada en el logro de determinados objetivos creativos o de ayuda a la toma de decisiones. Por lo tanto, un laboratorio virtual puede dedicarse prácticamente a todas las esferas de la actividad intelectual humana.” [3]

Esta investigación se apoya en la segunda definición citada, ya que el Laboratorio Virtual de Sistemas Operativos para el cual se modela el Subsistema de Ejercicios tiene entre sus objetivos apoyar el aprendizaje colaborativo y semipresencial en la universidad, para alcanzar un mayor rendimiento del nuevo modelo de formación.

No obstante es posible concretar que los laboratorios virtuales, en general, son medios accesibles y adaptables para poner en práctica conocimientos y representan una opción creativa, moderna y económica para instituciones universitarias que requieran laboratorios dentro de sus procesos de formación.

1.1.1 Tipos de laboratorios virtuales

Existen diversos tipos de laboratorios virtuales, pero generalmente se agrupan en tres clasificaciones generales:

Laboratorios virtuales software: Son laboratorios virtuales desarrollados como un software destinado a ejecutarse de forma independiente en la máquina del usuario y cuyos servicios no requieren de recursos externos.

Laboratorios virtuales web: Se basan en un software que depende de recursos ubicados en un servidor y al cual se accede mediante un navegador web desde cualquier máquina conectada a la red.

Laboratorios remotos: Estos laboratorios requieren de equipos servidores específicos que les den acceso a las máquinas a operar de forma remota, y no pueden ofrecer su funcionalidad ejecutándose de forma local.

El Laboratorio Virtual de Sistemas Operativos, para el cual se desea modelar el Subsistema de Ejercicios, clasifica entre los laboratorios virtuales de tipo web. Esta decisión es tomada a partir de la necesidad de integrarlo con la plataforma *Moodle* (utilizada en la UCI para la gestión de cursos) y centralizar las fuentes electrónicas de conocimiento, permitiendo que todos los usuarios de la institución tengan acceso a ella

desde diferentes ubicaciones y de forma simultánea. A diferencia del Laboratorio Virtual el Subsistema de Ejercicios debe ser diseñado como una aplicación de escritorio a solicitud del cliente y como requisito de su investigación a la cual tributa este trabajo.

1.1.2 Ventajas e inconvenientes de los laboratorios virtuales

Dentro de las ventajas más significativas del uso de laboratorios virtuales se destacan las siguientes:

- Acerca y facilita a un mayor número de alumnos la realización de experiencias, aunque alumno y laboratorio no coincidan en el espacio.
- El estudiante accede a los equipos del laboratorio a través de un navegador, pudiendo experimentar sin riesgo alguno, además se flexibiliza el horario de prácticas y evita la saturación por el solapamiento con otras asignaturas.
- Reducen el coste del montaje y mantenimiento de los laboratorios tradicionales, siendo una alternativa barata y eficiente, donde el estudiante simula los fenómenos a estudiar como si los observase en el laboratorio tradicional.
- La simulación en el laboratorio virtual, permite obtener una visión más intuitiva de aquellos fenómenos que en su realización manual no aportan suficiente claridad gráfica.
- Los estudiantes aprenden mediante prueba y error, sin miedo a sufrir o provocar un accidente, sin avergonzarse de realizar varias veces la misma práctica, ya que pueden repetirlas sin límite, sin temor a dañar alguna herramienta o equipo. [4]

Aparejados a estos beneficios también se presentan algunos inconvenientes:

- El laboratorio virtual no puede sustituir la experiencia práctica altamente enriquecedora del laboratorio tradicional. Ha de ser una herramienta complementaria para formar a la persona y obtener un mayor rendimiento.
- En el laboratorio virtual se corre el riesgo de que el alumno se comporte como un mero espectador.
- El alumno no utiliza elementos reales en el laboratorio virtual, lo que provoca una pérdida parcial de la visión de la realidad. Además, no siempre se dispone de la simulación adecuada para el tema que el profesor desea trabajar. [4]

1.2 Autoevaluación en entornos virtuales de enseñanza-aprendizaje

Antes de entrar en el término autoevaluación es necesario definir el concepto de evaluación como punto de partida y del cual se deriva.

Según la Real Academia Española *"la evaluación es la acción de estimar, apreciar, calcular o señalar el valor de algo"*. [5]

En la educación es un conjunto de actividades programadas para recoger información sobre la que profesores y alumnos reflexionan y toman decisiones para mejorar sus estrategias de enseñanza y aprendizaje e introducir en el proceso en curso las correcciones necesarias.

Una vez que el estudiante hace uso de estas actividades para verificar su propio progreso en el momento que considere y con los materiales que decida convenientes, es posible referirse a esta acción con el término de autoevaluación.

Para Gabriel Catari Padilla *"puede y debe ser un instrumento que facilite atender, respetar y valorar los distintos ritmos de aprendizaje según las diferentes características del alumno."* [6]

De las razones anteriormente expuestas no cabe duda que la autoevaluación del estudiante debe ser utilizada como estrategia para afrontar la diversidad de intereses, necesidades y ritmos de aprendizaje de los alumnos. Para atender a la diversidad es necesario utilizar diferentes instrumentos evaluativos que permitan valorar la progresión de las capacidades de cada alumno.

Existen diversos métodos tradicionales de evaluación. Entre los principales cabe destacar los exámenes de teoría, de problemas, orales, la realización de trabajos y proyectos y las pruebas de respuesta objetiva. En los entornos virtuales de enseñanza - aprendizaje sólo son implementadas de forma sencilla y con corrección automática, las **pruebas de respuesta objetiva**. El motivo de esta restricción reside en la dificultad de corrección, por parte de un ordenador, de preguntas que no tengan una respuesta objetiva.

Las principales ventajas que aporta el uso de un sistema de autoevaluación en los entornos virtuales son las siguientes: [7]

- Posibilita un seguimiento individualizado del aprendizaje del alumno.
- Permite evaluar conocimientos y habilidades.
- Facilita el establecimiento de una evaluación continuada durante el proceso de aprendizaje y reduce el tiempo empleado en su diseño, distribución y desarrollo.
- Agrega una gran flexibilidad temporal y espacial del sistema tanto para la configuración de ejercicios como de su realización. En este sentido puede ser especialmente útil para permitir que el alumno pueda seguir su propio ritmo de aprendizaje.
- Proporciona una respuesta inmediata (retroalimentación) de los resultados de los ejercicios.
- El almacenamiento de los resultados facilita la creación de informes y tratamiento de datos tanto a nivel de un alumno o de un grupo de alumnos como de las preguntas utilizadas.
- La base de datos de preguntas puede reutilizarse en otros cursos.
- La no necesidad de corregir por parte del profesor lo hace especialmente apropiado para grandes grupos de alumnos.

Un alto porcentaje de entornos virtuales de enseñanza - aprendizaje incluyen un sistema de pruebas de respuesta objetiva, de tipo cuestionario o autoevaluación mediante un módulo de software con acceso a una base de datos. Generalmente sus funcionalidades incluyen la creación de preguntas para este tipo de pruebas, la configuración de ejercicios, la generación de las pruebas, la gestión, corrección y almacenamiento de las respuestas, etc.

Las preguntas que componen las pruebas o cuestionarios pueden ser de tipo verdadero/falso, completar espacios vacíos, de respuesta única o múltiple, con solución numérica entera o real, preguntas de relación o emparejamiento, de ordenación, ejercicios de respuesta corta o de cadena de caracteres y otros cuya solución es proporcionada de forma gráfica, tablas, mapas de imágenes. Debe tenerse en cuenta también la posibilidad de incluir elementos multimedia en el enunciado como imágenes, gráficas, ecuaciones, sonidos, vídeos, etc.

En cuanto a la generación de las pruebas debe considerarse la capacidad de crearlas a partir de una base de datos de preguntas y la flexibilidad en su diseño con capacidad para controlar aspectos tales como:

puntuación, número de intentos, duración de la prueba, sin dejar a un lado la posibilidad de integrar el uso de emuladores o simuladores como parte de la evaluación que se va a efectuar.

Como ejemplo de entorno virtual que implemente un módulo de autoevaluación podemos mencionar la plataforma de tele-enseñanza AulaWeb, desarrollada en el departamento de Automática, Ingeniería Electrónica e Informática Industrial de la Escuela Técnica Superior de Ingenieros Industriales Universidad Politécnica de Madrid.

1.2.1 El módulo de autoevaluación del entorno AulaWeb

El entorno de la aplicación permite la realización de varias actividades docentes mediante el empleo de un ordenador con conexión a Internet y un navegador web independientemente de su sistema operativo. La interfaz de AulaWeb está basada en una estructura mixta de iconos y menús, que permite al usuario saber en todo momento como acceder a una determinada zona o módulo del sistema con el objeto de facilitar al máximo la navegación del usuario. La aplicación incluye un sistema de seguridad a nivel de usuario para el acceso de alumnos, profesores y administrador general que se realiza a través de la red y tres interfaces distintas según el tipo de usuario: alumno, profesor y administrador general. Mediante un entorno amigable y sencillo de utilizar, se trata de facilitar tanto al alumnado como al profesorado, tareas como la publicación y recogida de información y recursos formativos, la realización y entrega de trabajos y prácticas, la ejecución de actividades de autoevaluación o el establecimiento de tutorías telemáticas en tiempo real. [7]

Entre las funcionalidades que incluye el módulo de autoaprendizaje de AulaWeb se encuentran:

1. Un gestor de preguntas que facilita al profesor introducir preguntas en la base de datos de la aplicación, asignándole a cada una un capítulo del temario y un nivel de complejidad (muy fácil, fácil, medio, difícil y muy difícil). Las preguntas pueden ser de tipo verdadero/falso, selección simple, selección múltiple, respuesta numérica entera o real, respuesta de tipo cadena de caracteres y enunciado variable.

2. Un sistema de configuración que permite indicar los parámetros de los ejercicios.

- Título del ejercicio.
- Grupo de alumnos al que va dirigido.
- Número de preguntas.
- Tipo de preguntas.
- Capítulo.
- Nivel de dificultad de las preguntas.
- Tiempo límite (duración en pruebas cronometradas).
- Forma de corrección.
- Fecha límite de finalización.
- Máscara de red de los ordenadores conectados a internet desde los cuales se puede realizar el ejercicio.

3. El generador y administrador de ejercicios que, en base al contenido de la base de datos de preguntas y a los parámetros de configuración del ejercicio correspondiente, compone su contenido, presenta las preguntas y almacena las respuestas y los resultados en la base de datos y muestra dichos resultados a alumnos y profesores de la asignatura correspondiente.

1.2.2 Aporte a la solución

A partir del estudio anterior se obtuvo una aproximación a la solución que se desea, en la cual se incluyen algunas funcionalidades representativas que facilitarán la gestión de las evaluaciones de manera organizada y bien estructurada.

El Subsistema de Ejercicios propiciará autoevaluación mediante ejercicios y cuestionarios.

Los profesores podrán introducir y almacenar en el subsistema ejercicios, preguntas y cuestionarios, configurando para cada uno la información que lo caracteriza y por la cual podrán ser organizada las evaluaciones para facilitar la búsqueda. La información estará dada por:

- Nombre.

- Contenido al que pertenece.
- Complejidad (Baja, Media, Alta, Muy Alta).
- Tiempo de respuesta.
- Cantidad de intentos posibles a realizar.
- Objetivos y habilidades que desarrolla.

Las preguntas que conforman los cuestionarios serán creadas y almacenadas de forma independiente, lo cual permitirá reutilizarlas en otros tipos de evaluación que se incorporen posteriormente. Podrán ser de tipo:

- Verdadero/Falso.
- Selección única o múltiple.
- Relación o emparejamiento.
- Respuesta numérica entera o real y de cadena de caracteres.

Para realizar los ejercicios se contará con evaluadores que serán invocados por el subsistema en correspondencia con el tema del ejercicio.

1.3 Ingeniería de Requisitos

Los requisitos representan el pilar fundamental en el que se basa el desarrollo de software. Son las capacidades y/o cualidades que debe cumplir el futuro sistema a construir, las cuales son establecidas por el cliente de forma previa al comienzo de su desarrollo.

La Ingeniería de Requisitos (IR) es la disciplina dentro de la Ingeniería de Software que comprende todas las actividades que se realizan con el fin de guiar las necesidades del cliente durante todo el ciclo de desarrollo del software; permite que los requisitos alcancen un estado óptimo antes de arribar a la fase de diseño en el proyecto y que se obtenga un producto con mayor calidad.

Los principales beneficios que se obtienen de la IR son: [8]

- *Permite gestionar las necesidades del proyecto en forma estructurada:* Cada actividad de la IR consiste en una serie de pasos organizados y bien definidos.
- *Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados:* La IR proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios.
- *Disminuye los costos y retrasos del proyecto:* Muchos estudios han demostrado que reparar errores por un mal desarrollo no descubierto a tiempo, es sumamente caro; especialmente aquellas decisiones tomadas durante la IR.
- *Mejora la calidad del software:* La calidad en el software consiste en cumplir un conjunto de requisitos (funcionalidad, facilidad de uso, confiabilidad, desempeño, etc.).
- *Mejora la comunicación entre equipos:* La especificación de requisitos representa una forma de consenso entre clientes y desarrolladores. Si este consenso no ocurre, el proyecto no será exitoso.
- *Evita rechazos de usuarios finales:* La IR obliga al cliente a considerar sus requisitos cuidadosamente y revisarlos dentro del marco del problema, por esa razón se le involucra durante todo el desarrollo del proyecto.

1.3.1 Actividades y técnicas de la Ingeniería de Requisitos

Las actividades de la IR varían tanto en número como en nombres dependiendo del tamaño del proyecto y del modelo de proceso de software utilizado para el ciclo de desarrollo. Para cada una de estas propone un grupo de técnicas que pueden ser empleadas en combinación para establecer los requisitos exactos de las personas implicadas, con el objetivo de producir un sistema que resuelva las necesidades del negocio. De forma general las actividades son de 5 clases:

- Captura de Requisitos.
- Análisis de Requisitos.
- Especificación de Requisitos.
- Validación de Requisitos.
- Gestión de Requisitos.

A continuación se definen las actividades y técnicas que serán usadas para el desarrollo de este trabajo:

Captura de requisitos: Actividad mediante la cual se identifican las necesidades del cliente en un estudio del dominio del problema. Las técnicas a ser utilizadas para obtener las cualidades y características del subsistema serán:

- **Entrevistas:** Por lo general se entrevista a una selección de personas que represente a todos los sectores críticos de la organización, con el énfasis puesto en los sectores más afectados o que harán un uso más frecuente del nuevo sistema. Los requisitos que surgen de las entrevistas a menudo se contradicen unos a otros o se formulan desde la ignorancia de los detalles del funcionamiento del sistema, sus potencialidades, interdependencias o limitaciones; por lo que se debe trabajar con los mismos para corregir sus fallos.

- **Tormenta de ideas:** No es más que una reunión con un grupo reducido, que generalmente no exceden las diez personas, para que cada uno de los participantes exponga sus propias ideas acerca de las funciones que el sistema debe cumplir.

Especificación de requisitos: En esta actividad se representan y describen los requisitos capturados. Generalmente son empleadas la técnica de casos de uso para representarlos y las plantillas o patrones para describirlos.

- **Casos de uso:** Esta técnica consiste en graficar la relación del sistema con los usuarios u otros sistemas. El objetivo de esta práctica es mejorar la comunicación entre los usuarios y los desarrolladores, mediante la prueba temprana de prototipos para minimizar cambios hacia el final del proyecto y reducir los costos finales.

- **Plantillas o patrones:** Esta técnica, recomendada por varios autores, tiene por objetivo el describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la

ambigüedad del lenguaje natural al estructurar la información. Cuanto más estructurada sea esta menos ambigüedad ofrece.

Validación de requisitos: En esta actividad se comprueba que los requisitos implementados se corresponden con lo que inicialmente se pretendía. Para ello suelen usarse con mucha frecuencia las técnicas prototipos y matriz de trazabilidad.

- **Prototipos:** Un prototipo es una pequeña muestra, de funcionalidad limitada, de cómo sería el producto final una vez terminado. Ayudan a conocer la opinión de los usuarios y rectificar algunos aspectos antes de llegar al producto terminado. Pueden ser: diagramas o aplicaciones operativas con funcionalidades sintetizadas.

- **Matriz de trazabilidad:** Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo los objetivos que cubre cada requisito, de esta forma, se podrán detectar inconsistencias u objetivos no cubiertos.

1.3.2 Herramienta utilizada para la construcción de los prototipos

Axure RP es una herramienta para la construcción de prototipos de interfaz de usuario para sistemas computacionales. Genera un gran valor agregado a las actividades de captura y validación de requisitos debido a que está enfocado justamente en el punto más crítico de los proyectos de software, la satisfacción del cliente, logrando un excelente puente entre el análisis y el diseño de todo proyecto.

Proporciona:

- Construcción ágil de prototipos de interfaz de usuario.
- Soporte para los elementos comunes en estos prototipos (listas de selección, casillas de texto, entre otras).
- Navegación entre las páginas.
- Generación de prototipos en HTML.

1.4 Metodologías de desarrollo

Las metodologías guían el proceso de desarrollo de un software de manera organizada y planificada, en vista a lograr mayor eficiencia y obtener un resultado satisfactorio. Describen una serie de actividades, herramientas y técnicas a utilizar durante el ciclo de vida del software.

Disponer de metodologías diferentes para aplicar a proyectos que se desarrollen resulta imprescindible, de acuerdo a las necesidades cambiantes que tiene el entorno de desarrollo actual y el acelerado progreso de la informática a nivel mundial. Estas metodologías se clasifican en tradicionales y ágiles.

Las metodologías tradicionales están basadas básicamente en la división del proceso de desarrollo de software en etapas que se van a realizar de una manera secuencial y organizada. Las mismas exigen un alto grado de control y organización con los cambios, y generan además una cantidad notable de documentación.

En cambio las metodologías ágiles nacen con el objetivo de permitir al equipo desarrollar un software de forma rápida y que a la vez permita el control de los cambios que puedan ocurrir durante el desarrollo del software. Estas intentan evitar los caminos pesados de las metodologías tradicionales, enfocándose en las personas y los resultados.

1.4.1 Proceso Unificado de Rational

El Proceso Unificado de Rational (RUP por sus siglas en inglés) es una metodología tradicional para el desarrollo de software que constituye el estándar más utilizado para el análisis, implementación y documentación de sistemas orientados a objetos. Puede especializarse para una gran variedad de sistemas, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. Utiliza el Lenguaje Unificado de Modelado (UML por sus siglas en inglés) para preparar todos los esquemas de un sistema software, siendo UML una parte esencial del Proceso Unificado. Sus principales elementos son:

Trabajadores: Definen el comportamiento y roles de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo.

Actividades: Son realizadas por un trabajador y manipulan elementos.

Artefactos: Son los productos tangibles del proyecto, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

Flujo de actividades: Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

El ciclo de vida de RUP se caracteriza por ser:

- ***Dirigido por casos de uso:*** Significa que el proceso de desarrollo sigue un hilo que avanza a través de una serie de flujos de trabajo que parten de los casos de uso. Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.

- ***Centrado en la arquitectura:*** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. La arquitectura y los casos de uso deben evolucionar en paralelo.

- ***Iterativo e incremental:*** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

El proceso unificado se divide en ciclos de trabajo, teniendo un producto superior como resultado de cada ciclo. Estos se componen en su interior por varias fases, en las cuales se llevan a cabo un conjunto de flujos para el desarrollo de todo el proyecto.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. [9]

1.4.2 Programación Extrema

La programación extrema o XP (acrónimo de *Extreme Programming*) se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar pura lógica. Los objetivos de XP son muy simples: la satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, se debe responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación. El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software. [10]

Esta metodología es aplicada por su rapidez en proyectos a corto plazo. Como su nombre lo indica se centra en la programación y presenta como peculiaridad que incluye al cliente como parte del equipo de desarrollo.

1.4.3 SCRUM

SCRUM es un proceso en el que se aplican ciertas prácticas para mejorar sosteniblemente el trabajo en equipo, logrando obtener el mejor resultado posible de un proyecto. Regularmente se realizan entregas parciales del producto final, donde se van incorporando las diferentes funcionalidades, priorizadas siempre por el beneficio que aportan al cliente.

Está especialmente indicada para proyectos en entornos complejos, que necesiten obtener resultados pronto, con requisitos cambiantes o vagamente definidos, donde la improvisación, la competitividad, la flexibilidad y la productividad son esenciales. También se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado o la calidad no es

aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

En *Scrum* un proyecto se ejecuta en bloques temporales cortos y fijos, cuya duración puede variar entre las 2 y las 4 semanas. Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando este lo solicite. [11]

1.5 Lenguajes de modelado

Un Lenguaje de Modelado es un conjunto estandarizado de símbolos y modos de disponerlos para modelar un diseño de software orientado a objetos o parte de él. En la actualidad son muy utilizados el Lenguaje Unificado de Modelado y la Notación de Modelado de Procesos de Negocio.

1.5.1 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML, acrónimo de *Unify Model Process*) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la implementación y configuración con los diagramas de despliegue. [12] Es el estándar más ampliamente reconocido para la especificación, diagramación y documentación de software de calidad.

Los beneficios que se consiguen al utilizar UML son varios, por un lado el uso de lenguajes visuales facilitan su asimilación y entendimiento por parte del equipo de desarrollo; el tiempo invertido en el desarrollo de la arquitectura se minimiza; la detección y resolución de errores se agiliza siempre y cuando se haga uso de herramientas adecuadas de diagnóstico y depuración; y la trazabilidad y documentación del proyecto se realiza de una forma ordenada y guiada por los casos de uso. Pero si hay una ventaja que

destaca sobre todas las demás es la notable efectividad y productividad que se consigue en labores de diseño arquitectónico y mantenimiento haciendo uso de UML frente a la realización de las mismas tareas en ausencia de modelos. [13]

1.5.2 Notación de Modelado de Procesos de Negocio

La Notación de Modelado de Procesos de Negocio (BPMN, acrónimo de *Business Process Modeling Notation*) es una notación gráfica para el modelado conceptual de procesos de negocio. Proporciona la capacidad de entender y definir procesos de negocio, tanto internos como externos, a través de un diagrama de procesos de negocio. Es un estándar de facto que provee una representación gráfica (mediante diagramas) para expresar los procesos de una empresa. Se diseñó con el objetivo de facilitar la comprensión por parte de todos los implicados (expertos TIC, analistas de negocio, directivos, etc.) que participan en el proceso. Un modelo de proceso de negocio describe cómo funciona el negocio, indicando las actividades involucradas en el mismo y la relación existente entre ellas. El modelado de procesos del negocio es empleado para la captura, y documentación y rediseño de procesos del negocio. [14]

1.6 Herramientas CASE

Las herramientas CASE (acrónimo de *Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) son aplicaciones informáticas que intentan proporcionar ayuda automatizada a las actividades del proceso de software. Permiten a los desarrolladores modelar y documentar sus artefactos, cubriendo el ciclo de vida del proceso de desarrollo de software.

1.6.1 Rational Rose Enterprise Edition

Herramienta CASE desarrollada por *Rational Corporation* que permite crear los diagramas que se van generando durante el proceso de ingeniería en el desarrollo del software. Posibilita además que el equipo

de desarrollo entienda mejor el problema, que identifique las necesidades del cliente en forma más efectiva y comunique la solución propuesta de forma más clara.
Entre las características principales de Rational se pueden destacar:

- Admite como notaciones: UML, OMT y Booch.
- Permite desarrollo multiusuario.
- Genera documentación del sistema.
- Disponible en múltiples plataformas.

1.6.2 Visual Paradigm para UML

Visual Paradigm para UML es una herramienta profesional que facilita el modelado del ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML soporta las últimas versiones del mismo y la Notación y Modelado de Procesos de Negocios. Ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste ya que permite generar código desde diagramas y generar documentación. [15]

Visual Paradigm ofrece distintas funcionalidades como:

- Entorno de creación de diagramas para UML 2.0.
- Diseño centrado en casos de uso y enfocado al negocio generando un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa en su versión profesional, e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDE.
- Disponibilidad en múltiples plataformas (Windows, Linux, etc.)

1.6.3 Enterprise Architect

Enterprise Architect es una herramienta multiusuario, compatible con el análisis y diseño UML y el modelado de procesos BPMN. Opera sobre el sistema operativo Windows y posee un ambiente rápido, fácil y flexible. Ofrece la posibilidad de rastrear las dependencias de soportes existentes, soporta la generación e ingeniería inversa de código fuente para varios lenguajes como: C++, C#, Java, Delphi, VB.Net, Visual Basic y PHP, además presenta un editor de código y genera informes con una alta calidad.

1.7 Selección de metodología, lenguaje y herramienta a utilizar

En los epígrafes anteriores se realiza un estudio de algunas metodologías, lenguajes y herramientas de modelado para el desarrollo de software con el objetivo de decidir cuál se ajusta a las necesidades del proyecto y facilitar el desarrollo de la solución.

Teniendo en cuenta que los analistas actuales no serán los que implementen el subsistema final, y que por ello el cliente exige la mayor claridad y detalle en los artefactos generados antes de entregar a los desarrolladores, se selecciona la metodología RUP que permitirá documentar de forma organizada y bien estructurada el análisis y diseño del subsistema que se desea modelar evitando resultados no deseados.

Como lenguaje de modelado se decide utilizar UML, el cual permitirá construir y documentar los artefactos del subsistema con una notación gráfica expresiva que facilitará su asimilación y entendimiento por parte del equipo de desarrollo.

Conociendo que es multiplataforma y la universidad posee licencia para su uso legal, se selecciona la herramienta CASE Visual Paradigm for UML, que soporta el lenguaje de modelado escogido y facilitará la diagramación visual y el diseño de la solución propuesta. Es importante resaltar que su capacidad de generar código y soportar el lenguaje Java permitirá un ahorro de tiempo y esfuerzo en la fase de implementación.

Conclusiones

El análisis de las funcionalidades frecuentemente implementadas en los entornos virtuales de enseñanza – aprendizaje permitió obtener una aproximación de la solución que se desea. Con el estudio de las actividades de la ingeniería de requisitos y sus técnicas se realiza una selección de las más efectivas, dadas las peculiaridades del negocio. Como resultado del estudio de diferentes metodologías de desarrollo, lenguajes y herramientas de modelado se determina el uso de las más apropiadas para el desarrollo del subsistema.

Capítulo 2. Descripción de la solución propuesta

Introducción

El presente capítulo se dedica a la descripción de la solución propuesta para el problema que motiva esta investigación. Son representadas las clases conceptuales del negocio en cuestión, a través del Modelo de Dominio, con el objetivo de definir conceptos claves para establecer un lenguaje común entre los involucrados. Se especifican los requisitos y se construye el diagrama de casos de uso del sistema que representa la relación entre los actores que interactúan con el sistema y los casos de uso que representan las funcionalidades a ser automatizadas. Una vez precisadas las funcionalidades se construye el modelo de análisis para establecer la relación entre clases y objetos del dominio, especificando en el diseño los métodos que permiten automatizar cada funcionalidad del subsistema.

2.1 Modelo de Dominio

Existen varias alternativas para llevar a cabo el modelamiento de un sistema. RUP, como metodología de desarrollo, propone la realización de un modelo de negocio para el caso en el que los procesos dentro del entorno estén claramente definidos, y la realización de un modelo de dominio para los escenarios en los que no puedan identificarse tales procesos del negocio.

El Modelo de Dominio es una representación de clases conceptuales u objetos del mundo real, significativos en el problema o área de interés. Se utiliza para establecer cómo funciona el ámbito del problema en un lenguaje común para todos los implicados en el desarrollo.

2.1.1 Conceptos del modelo de dominio

Es necesario tener un vasto conocimiento de cómo debe funcionar el proceso en cuestión, para poder capturar correctamente los requisitos y así poder construir un sistema con las características que el cliente desea. Este modelo va a contribuir posteriormente a identificar algunas clases que se utilizarán en el sistema. Primeramente se identificarán todos los conceptos que se utilizarán en el diagrama:

Concepto	Descripción
Evaluador	Sistema que a partir de los datos de entrada, chequea y evalúa las respuestas del estudiante al realizar un ejercicio.
Subsistema de Ejercicios.	Subsistema en el que se gestionan los ejercicios y cuestionarios para evaluar a los estudiantes.
Ejercicios.	Colección de ejercicios disponibles para que el estudiante ponga en prácticas sus conocimientos.
Preguntas.	Colección de preguntas creadas por el profesor para ser incluidas luego en los cuestionarios u otro tipo de evaluación.
Cuestionarios.	Tipo de evaluación que contiene preguntas formuladas por el profesor y son resueltos por el estudiante.

Tabla 1. Conceptos del Modelo de Dominio.

A continuación se muestra el Modelo de Dominio correspondiente:

MODELO DE DOMINIO

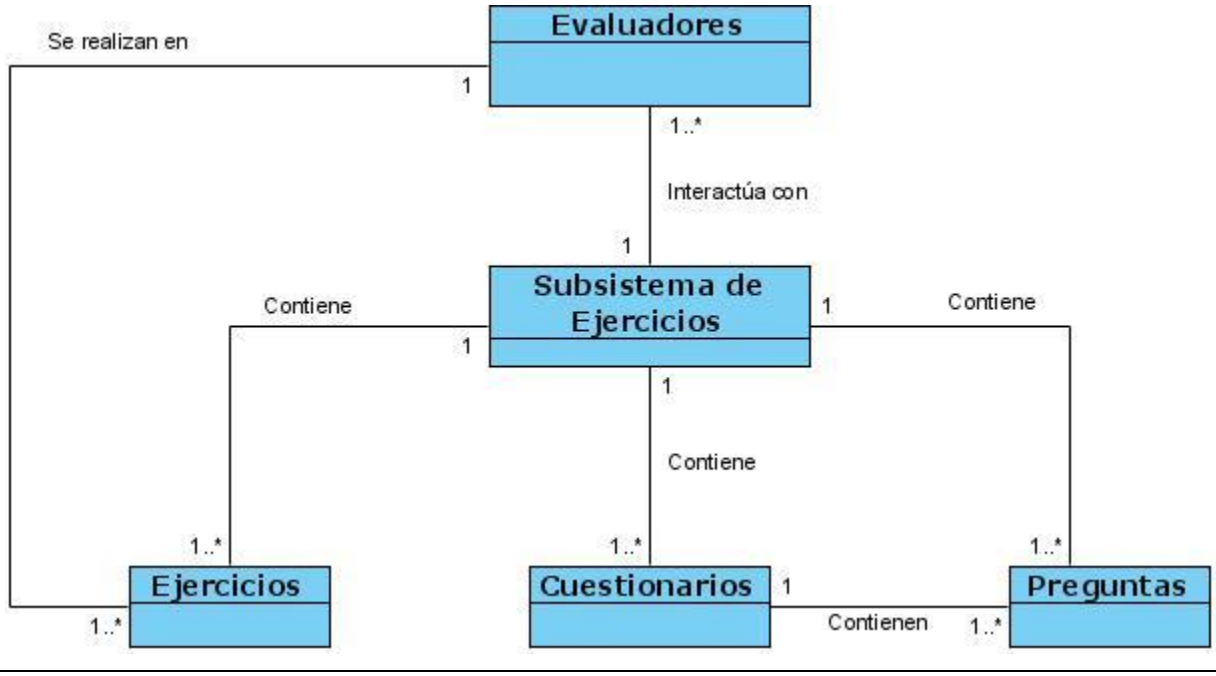


Figura 1. Modelo de Dominio.

2.2 Reglas de negocio

Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto del negocio. Para el Subsistema de Ejercicios se definen las siguientes reglas:

- Las evaluaciones deben estar separadas, atendiendo al tipo, en ejercicios o cuestionarios y a su vez ordenadas teniendo en cuenta el contenido al que pertenece, la complejidad, objetivos y habilidades que desarrolla.
- Se debe establecer el número de intentos para realizar los cuestionarios y la nota que se proponga puede ser de dos tipos: con penalización o la más alta.

- Al realizar cada evaluación debe ser registrada en la base de datos, incluyendo la nota, el tiempo consumido y el número de intentos.
- Al culminar la evaluación se debe enviar una confirmación de la misma al Módulo de Evaluación y Seguimiento y al de Autoevaluación, especificando el tema al que pertenece.

2.3 Especificación de requisitos del software

La especificación de requisitos de software es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye los casos de uso que describen las interacciones de los usuarios con el software y las cualidades o restricciones de diseño que el cliente desea para el producto final. Luego de la entrevista con el cliente y la tormenta de ideas realizada con un grupo de especialistas de la asignatura de Sistemas Operativos de la Facultad 3, personal involucrado en el dominio del problema, se obtuvieron los requisitos que a continuación se especifican.

2.3.1 Requisitos funcionales

Un requisito funcional (RF) define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas. Son condiciones o capacidades que el sistema debe cumplir. El subsistema de Ejercicios debe ser capaz de:

RF1: Cargar ejercicio.
RF2: Crear pregunta.
RF3: Modificar pregunta.
RF4: Eliminar pregunta.
RF5: Crear cuestionario.
RF6: Modificar cuestionario.
RF7: Eliminar cuestionario.
RF8: Habilitar ejercicio.
RF9: Habilitar cuestionario.

RF10: Deshabilitar ejercicio.
RF11: Deshabilitar cuestionario.
RF12: Realizar cuestionario.
RF13: Guardar progreso del cuestionario.
RF14: Realizar ejercicio.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente están vinculados a requerimientos funcionales, es decir una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. Las cualidades que deben caracterizar el subsistema son:

Interfaz de usuario:
RNF 1: La aplicación deberá poseer una interfaz amigable e intuitiva.
RNF 2: Los elementos gráficos en la interfaz deberán representar claramente la acción o información al que están asociados, de forma que la interfaz permita la interacción natural del sistema con sus usuarios.
RNF 3: El usuario debe poder acceder a los documentos de ayuda desde cualquier parte del sistema.

Seguridad:
RNF 4: El sistema debe poseer mecanismos de seguridad que garanticen la confidencialidad de la información basados en el modelo (Autentificación, Autorización y Auditoría).
RNF 5: Garantizar que la información sólo sea modificada por las personas que tienen permisos para realizar esta actividad.
RNF 6: Incluir mecanismo de recuperación de datos (salvas automáticas o copias de respaldo) que permitan recuperar la información ante un fallo del sistema.
RNF 7: Los mecanismos de seguridad no afectarán el tiempo de respuesta a las solicitudes de los usuarios.

Soporte:

RNF 8: El sistema debe ser capaz de asimilar la incorporación de nuevos recursos que enriquezcan su contenido.

Rendimiento:

RNF 9: El tiempo de respuesta del sistema no deberá exceder los 9 segundos.

Software:

RNF 10: Máquina Virtual Java.

RNF 11: Sistema Operativo Windows 98 en adelante.

Hardware:

RNF 12: Procesador a frecuencia de 700 MHz como mínimo.

RNF 13: 256 MB de memoria RAM como mínimo.

2.3.3 Actores del sistema

Los actores del sistema representan el rol que juega una o varias personas, un equipo o un sistema automatizado. Son parte del sistema y pueden intercambiar información con él o ser recipientes pasivos de información. En este caso los actores que interactúan con el subsistema se definen en la siguiente tabla.

Actor	Justificación
Estudiante	Su interacción con el sistema sucede cuando realiza cuestionarios o ejercicios.
Profesor	Es el encargado de gestionar los cuestionarios y ejercicios que el estudiante debe tener disponibles para autoevaluar su aprendizaje.

Tabla 2. Actores del sistema.

2.3.4 Diagrama de Casos de Uso del Sistema

Un Diagrama de Casos de Uso del Sistema (DCUS) muestra la relación entre actores y casos de uso que representan las funcionalidades del sistema, en lo que se refiere a su interacción externa. De acuerdo a los requisitos obtenidos se identificaron las funcionalidades que se muestran en el siguiente DCUS en relación con los actores que inicializan la acción.

DIAGRAMA DE CASOS DE USO DEL SISTEMA

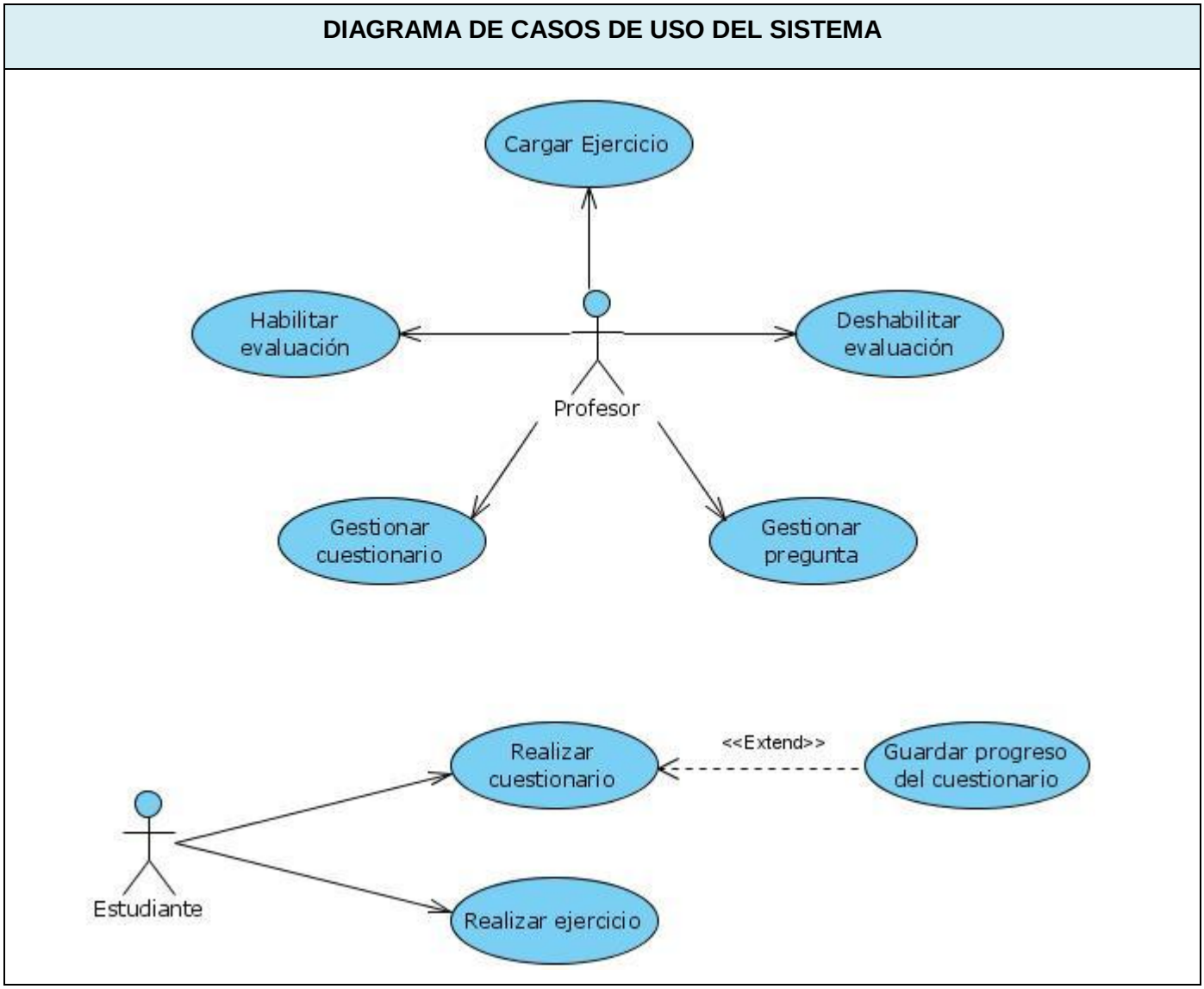


Figura 2. Diagrama de casos de uso del sistema.

2.3.4.1 Patrones de casos de uso aplicados

Los patrones de casos de uso se utilizan como técnicas o herramientas, resultantes de la experiencia de desarrolladores, que nos permiten de manera ágil resolver problemas que se presentan en la modelación de sistemas, obteniendo modelos de mayor calidad de forma más rápida.

Para modelar el DCUS se aplicó el patrón ***CRUD*** (acrónimo de *Creating, Reading, Updating, Deleting,* traducido en Crear, Leer, Actualizar y Eliminar) que consiste en fusionar casos de usos simples en una unidad conceptual nombrada Gestionar Información. Este patrón puede presentarse Parcial o Completo. Se dice que ha sido usado parcialmente cuando sólo se necesita gestionar dos de estas operaciones simples, y completo cuando se gestionan todas o al menos tres de ellas. En este caso se aplicó CRUD Completo para agrupar las funcionalidades de crear, modificar y eliminar tanto las preguntas como los cuestionarios, obteniéndose como resultado los casos de uso “Gestionar pregunta” y “Gestionar cuestionario”.

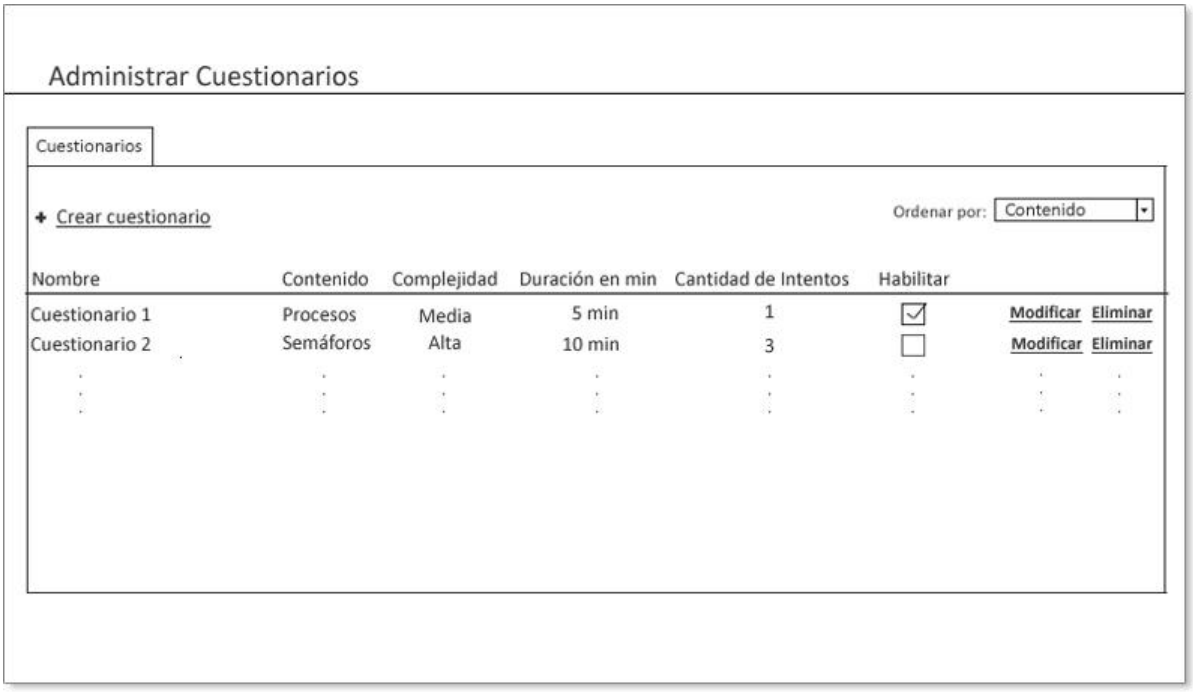
Otro patrón utilizado fue la **Concordancia - Adición** que se presencia cuando de un caso de uso base se extiende una operación que debe ser ejecutada como otro flujo dentro de este. En el diagrama se evidencia su aplicación en el caso de uso “Realizar cuestionario” que extiende la operación “Guardar progreso del cuestionario".

2.3.5 Descripción de los casos de uso del sistema

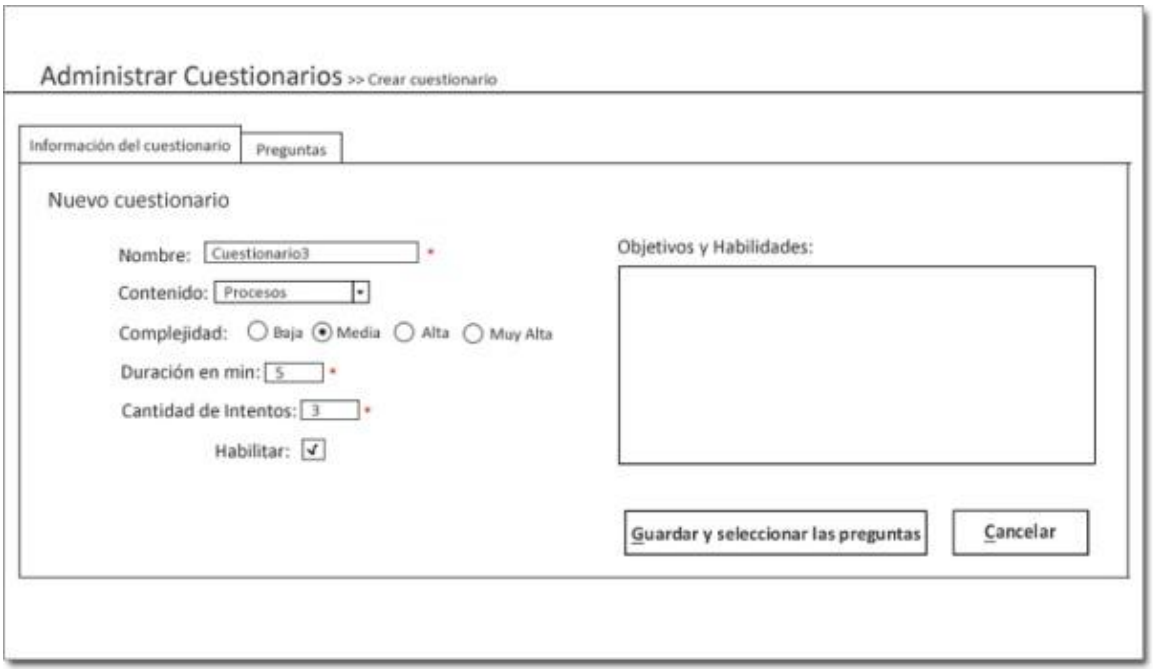
Las descripciones de casos de uso son reseñas textuales del caso de uso que explican el flujo de los procesos o actividades que tienen lugar en el mismo. A continuación se describe un caso de uso crítico del subsistema en cuestión, los demás se pueden encontrar en el Anexo 1.

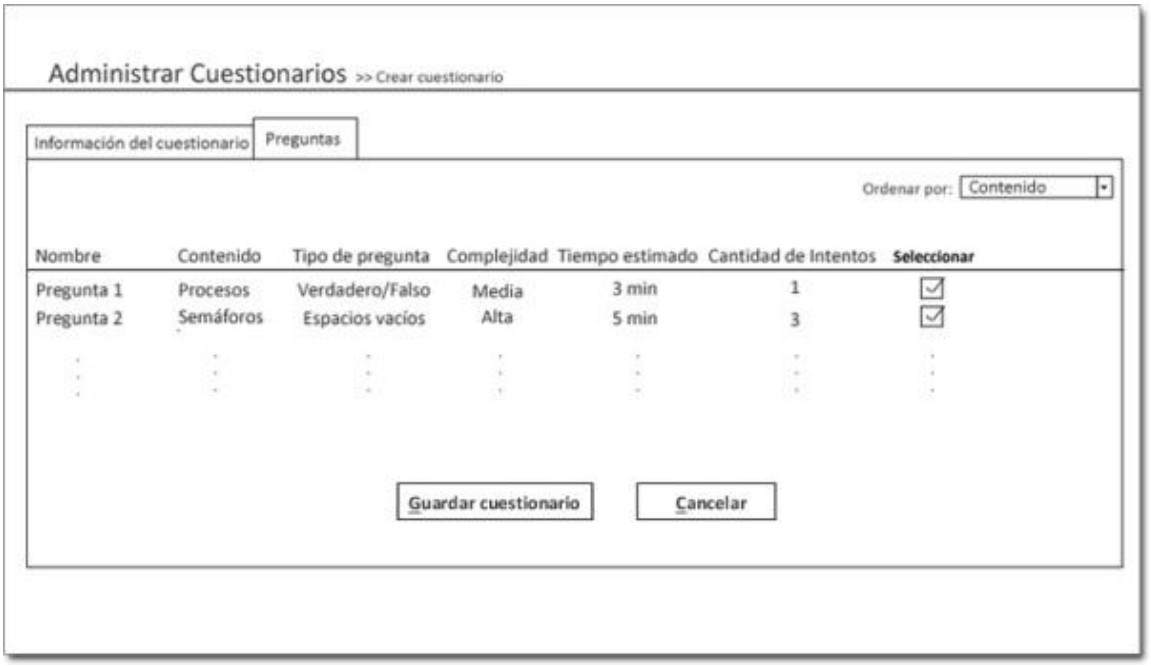
Descripción del Caso de Uso: "Gestionar cuestionario"

Caso de uso	Gestionar cuestionario.
Actores	Profesor.
Resumen	El caso de uso se inicia cuando el profesor decide crear, modificar o eliminar un cuestionario. El sistema muestra la interfaz correspondiente a la selección del profesor y termina cuando éste decide guardar cualquier cambio realizado.
Precondiciones	El profesor debe estar autenticado.
Poscondiciones	
Referencias	RF5, RF6, RF7.
Casos de uso asociados	
Prioridad	Crítico.
Flujo Normal de Eventos	
Sección: "General"	
Acción del actor	**Respuesta del sistema**
1. El profesor selecciona en el Menú Principal la opción "Administrar Cuestionarios".	1.1 El sistema muestra la interfaz correspondiente a "Administrar Cuestionarios" con el listado de cuestionarios existentes en el sistema y un grupo de opciones para su administración: - Crear cuestionario. - Modificar cuestionario. - Eliminar cuestionario.

2. El profesor selecciona una de las opciones de administración.	2.1 El sistema muestra la interfaz correspondiente a la opción seleccionada. - Si selecciona “Crear cuestionario”, ver Sección **“Crear cuestionario”**. - Si selecciona “Modificar cuestionario”, ver Sección **“Modificar cuestionario”**. - Si selecciona “Eliminar cuestionario”, ver Sección **“Eliminar cuestionario”**.
Prototipo no funcional	
	
Sección: “Crear cuestionario”	
Acción del actor	**Respuesta del sistema**

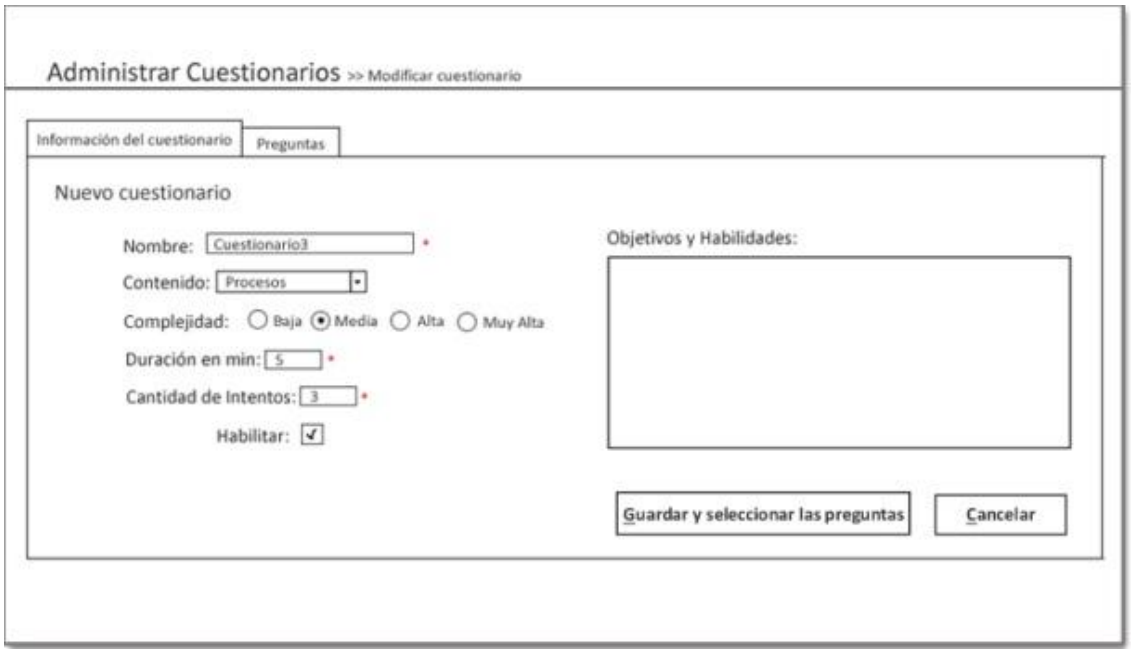
3. El profesor selecciona la opción “Crear cuestionario”.	3.1 El sistema muestra la interfaz correspondiente a “Crear cuestionario” y solicita los siguientes datos del cuestionario: - Nombre. - Contenido. - Complejidad (Baja, Media, Alta, Muy Alta). - Duración en minutos. - Cantidad de intentos. - Habilitar (Casilla de verificación) - Objetivos y Habilidades.
4. El profesor inserta los datos solicitados.	4.1 El sistema verifica que los datos insertados sean correctos y los campos obligatorios no estén vacíos.
	4.2 Si la verificación es correcta el sistema crea el nuevo cuestionario.
	4.3 El sistema muestra una lista de preguntas posibles a incluir en el cuestionario.
5. El profesor selecciona las preguntas que desea incluir en el cuestionario y oprime la opción “Guardar cuestionario”.	5.1 El sistema actualiza el nuevo cuestionario con las preguntas.
	5.2 El sistema regresa hacia la interfaz “Administrar Cuestionarios”.
Prototipo no funcional	

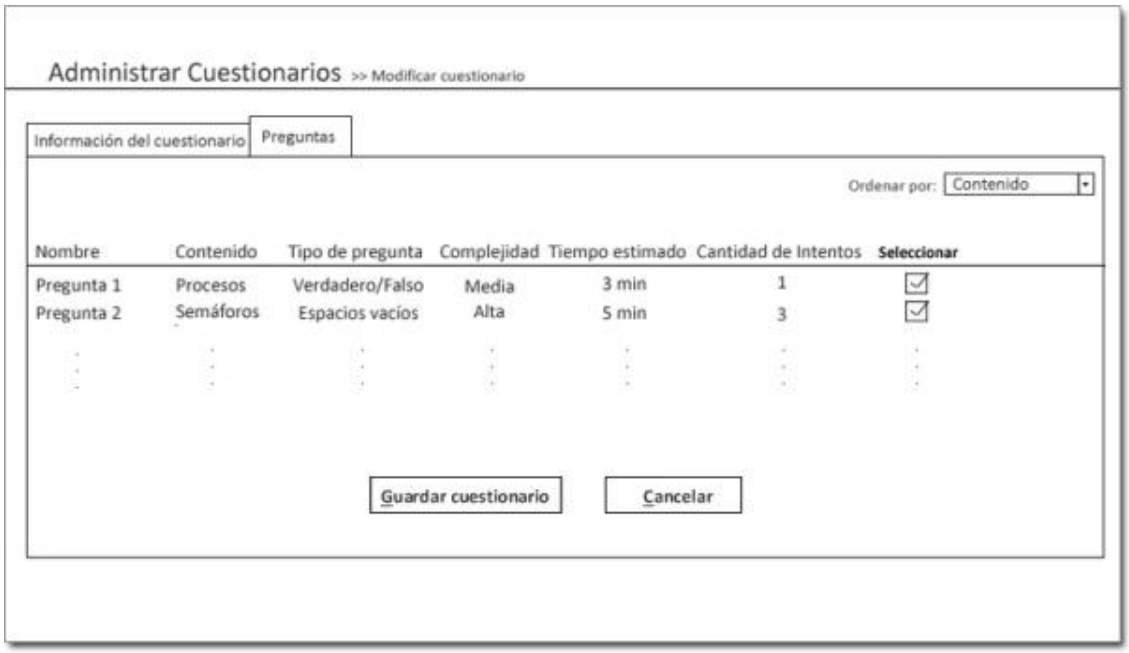




Sección: "Modificar cuestionario"

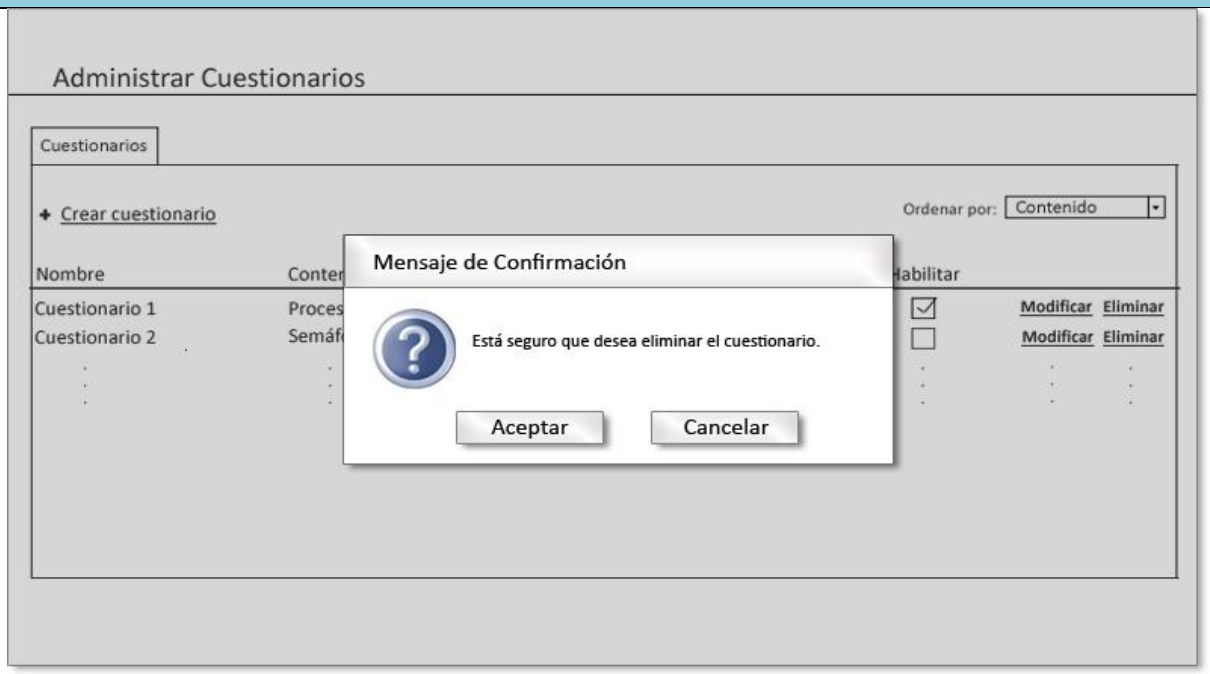
Acción del actor	Respuesta del sistema
3. El profesor selecciona la opción “Modificar cuestionario”.	3.1 El sistema muestra la interfaz correspondiente a “Modificar cuestionario” con los datos del cuestionario a modificar.
4. El profesor realiza las modificaciones que crea necesarias.	4.1 El sistema verifica que los datos modificados sean correctos.
	4.2 Si la verificación es correcta el sistema actualiza los datos del cuestionario con las modificaciones.
	4.3 El sistema muestra las preguntas del cuestionario seguida de la lista de preguntas existentes en el subsistema, para el caso que desee excluir e incluir otra.
5. El profesor selecciona las preguntas que conformarán el cuestionario y guarda los cambios.	5.1 El sistema actualiza el cuestionario con las preguntas seleccionadas.
	5.2 El sistema regresa hacia la interfaz “Administrar Cuestionarios”.
Prototipo no funcional	





Sección: "Eliminar cuestionario"

Acción del actor	Respuesta del sistema
3. El profesor selecciona la opción “Eliminar cuestionario”.	3.1 El sistema muestra un mensaje para que el usuario confirme su decisión, en caso positivo se elimina el cuestionario seleccionado.
	3.2 El sistema regresa hacia la interfaz “Administrar Cuestionarios”.



Flujos Alternos	
Sección: “Crear cuestionario”	
Acción del actor	**Respuesta del sistema**
4. El profesor inserta los datos dejando campos obligatorios en blanco o con errores.	4.1. El sistema muestra un mensaje: “Existen campos obligatorios vacíos o con errores”. Ir al paso 4 Sección: “Crear cuestionario”.
Sección: “Modificar cuestionario”	

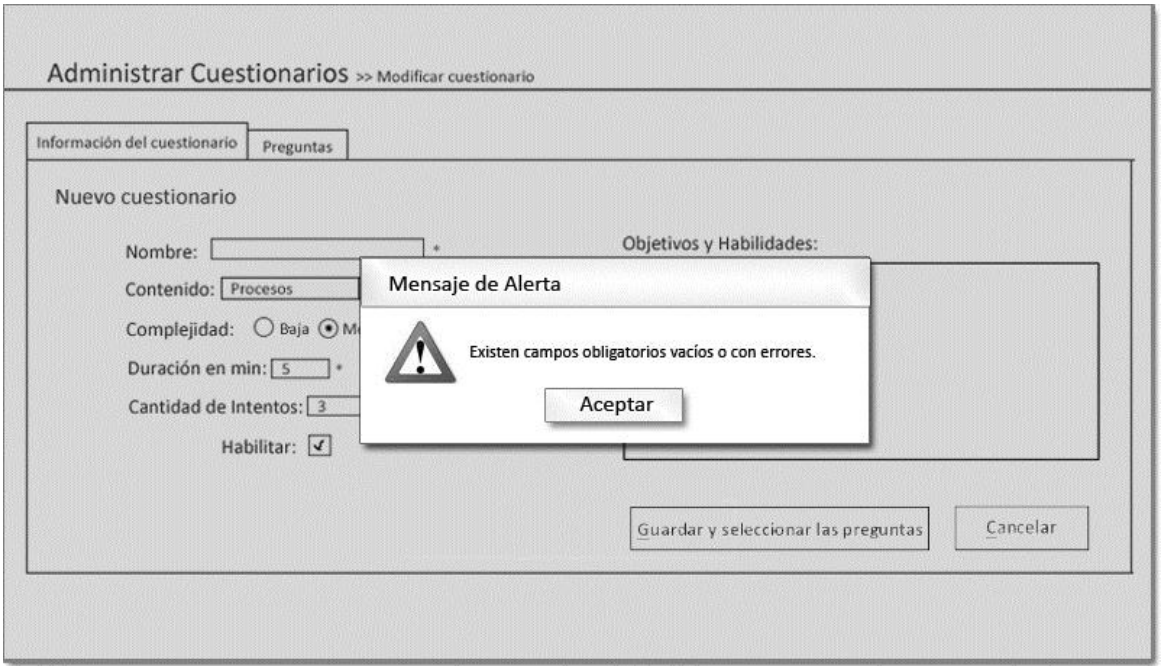
Acción del actor	Respuesta del sistema
4. El profesor llena los datos dejando campos obligatorios en blanco o con errores.	4.1. El sistema muestra un mensaje: “Existen campos obligatorios vacíos o con errores”. Ir al paso 4 Sección: “Modificar cuestionario”.
Prototipo no funcional	
	

Tabla 3. Descripción del CU Gestionar cuestionario.

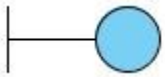
2.4 Modelo de Análisis

El Modelo de Análisis tiene la finalidad de refinar los casos de uso más detalladamente y realizar una retribución inicial del comportamiento del sistema a un conjunto de objetos que suministren el funcionamiento esperado. Es un modelo de objetos que describe la realización de casos de uso, y sirve como idealización para el modelo de diseño.

2.4.1 Diagramas de Clases de Análisis

El Diagrama de Clases de Análisis (DCA) está compuesto por clases de análisis y sus relaciones. Estas clases se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Se clasifican en:

Clase Interfaz:

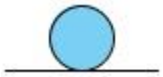


Se utiliza para modelar la interacción entre el sistema y sus actores lo que implica recibir y representar informaciones y peticiones de los usuarios y sistemas externos.

Clase Control:

Coordinan el trabajo de uno o unos pocos casos de uso, coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso, por lo que definen el flujo de control y las transacciones dentro de un caso de uso delegando el trabajo a otros objetos.

Clase Entidad:



Modelan información que posee una larga vida y que a menudo es persistente, además de fenómenos, conceptos y sucesos que ocurren en el mundo real.

A continuación se muestra un DCA del subsistema de Ejercicios, los restantes se pueden encontrar en el Anexo 2 de este documento.

Diagrama de clases de análisis. CUS "Gestionar cuestionario"

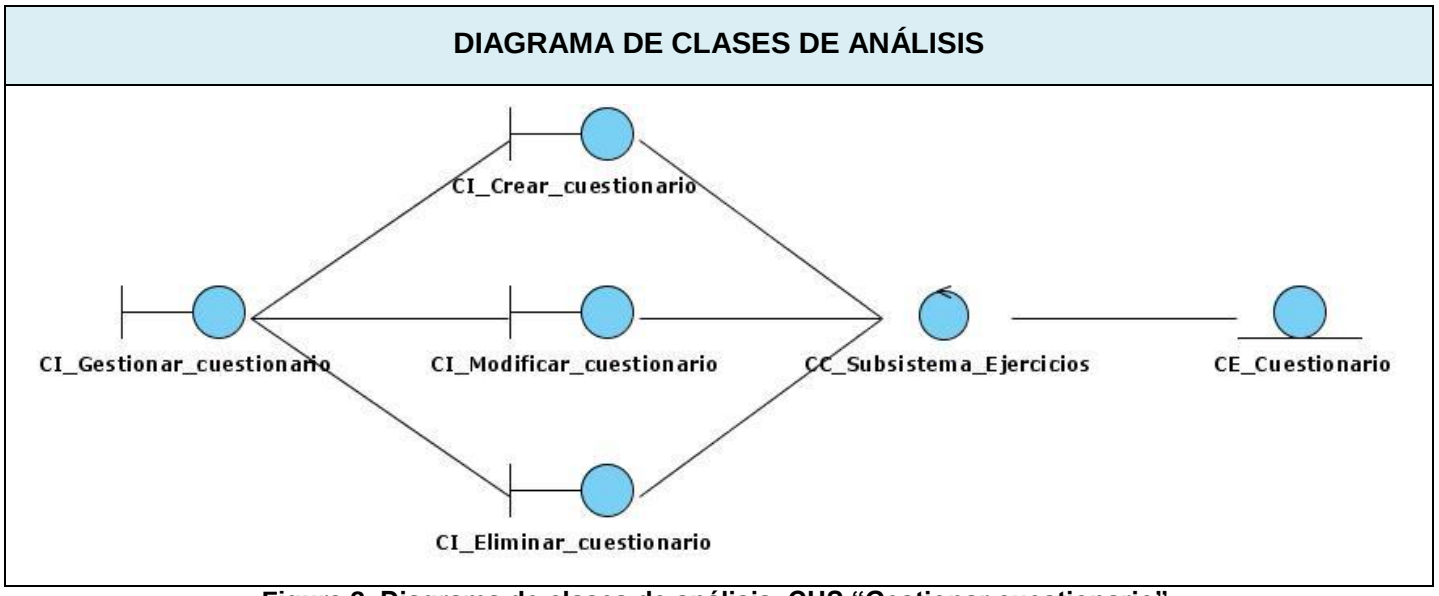


Figura 3. Diagrama de clases de análisis. CUS "Gestionar cuestionario".

2.5 Modelo de Diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. El modelo de diseño sirve de abstracción a la implementación y se utiliza como una entrada fundamental de las actividades de implementación. En la etapa de diseño ya se debe conocer el lenguaje de programación que será usado para implementar la aplicación. Los arquitectos del Laboratorio Virtual, luego de un estudio realizado, decidieron que se utilizará el lenguaje Java para su implementación.

2.5.1 Diagramas de Interacción

Los diagramas de interacción describen a través de mensajes la manera en que colaboran los objetos para cumplir ciertas tareas. Las interacciones proveen un comportamiento y típicamente implementan un

caso de uso. En UML existen dos tipos de diagramas de interacción: diagramas de secuencia y diagramas de colaboración. Estos últimos destacan la organización de los objetos que participan en una interacción.

A continuación se muestra uno de los diagrama de colaboración modelados para representar las interacciones entre los objetos que forman parte del subsistema. El resto de los diagramas se pueden encontrar en el Anexo 3.

Diagrama de Colaboración. CU "Gestionar cuestionario"- Sección: Crear cuestionario

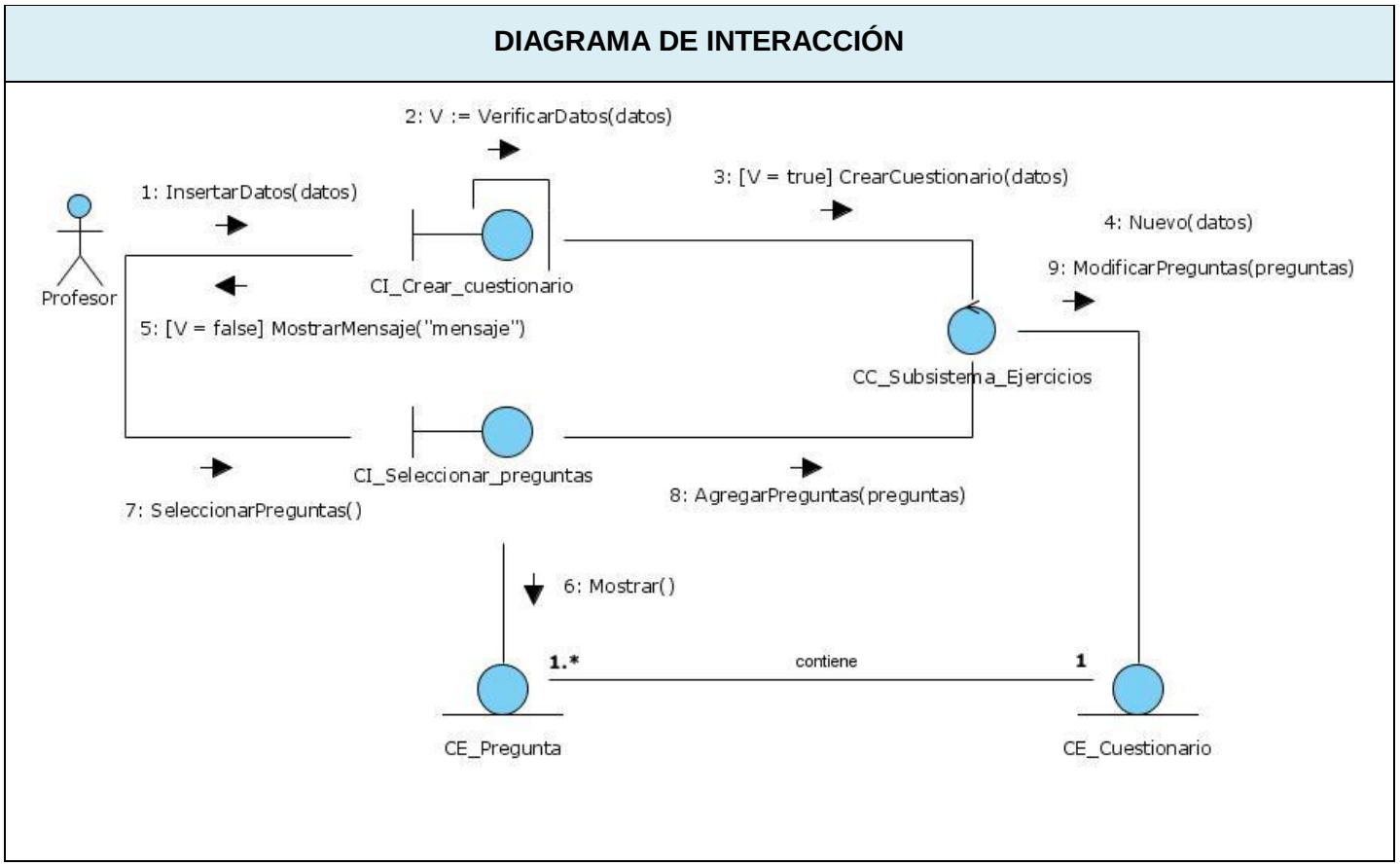


Figura 4. Diagrama de Colaboración. CU "Gestionar cuestionario"- Sección: Crear cuestionario.

Diagrama de Colaboración. CU "Gestionar cuestionario"- Sección: Modificar cuestionario

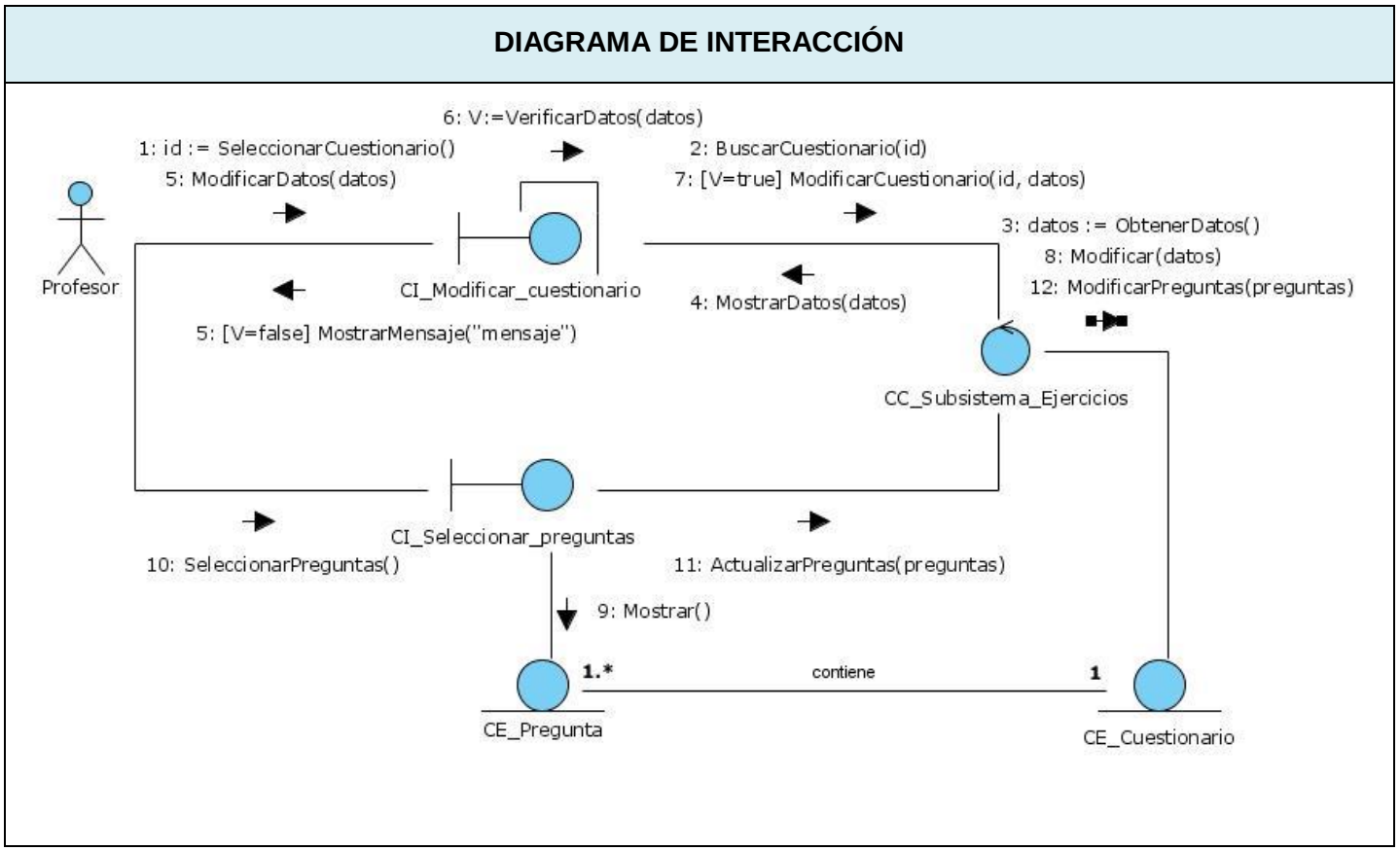


Figura 5. Diagrama de Colaboración. CU "Gestionar cuestionario"- Sección: Modificar cuestionario.

Diagrama de Colaboración. CU "Gestionar cuestionario"- Sección: Eliminar cuestionario

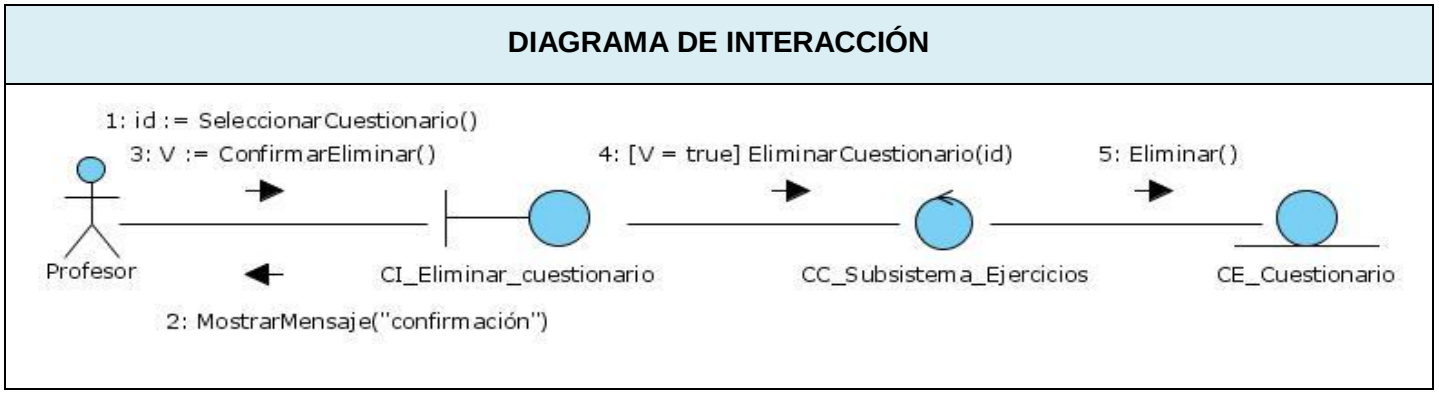


Figura 6. Diagrama de Colaboración. CU "Gestionar cuestionario"- Sección: Eliminar cuestionario.

2.5.2 Diagramas de Clases de Diseño

Los Diagramas de Clases de Diseño (DCD) describen gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. A continuación se muestra el diagrama elaborado a partir de las clases del análisis y las operaciones que se describen en los diagramas de colaboración antes presentados.

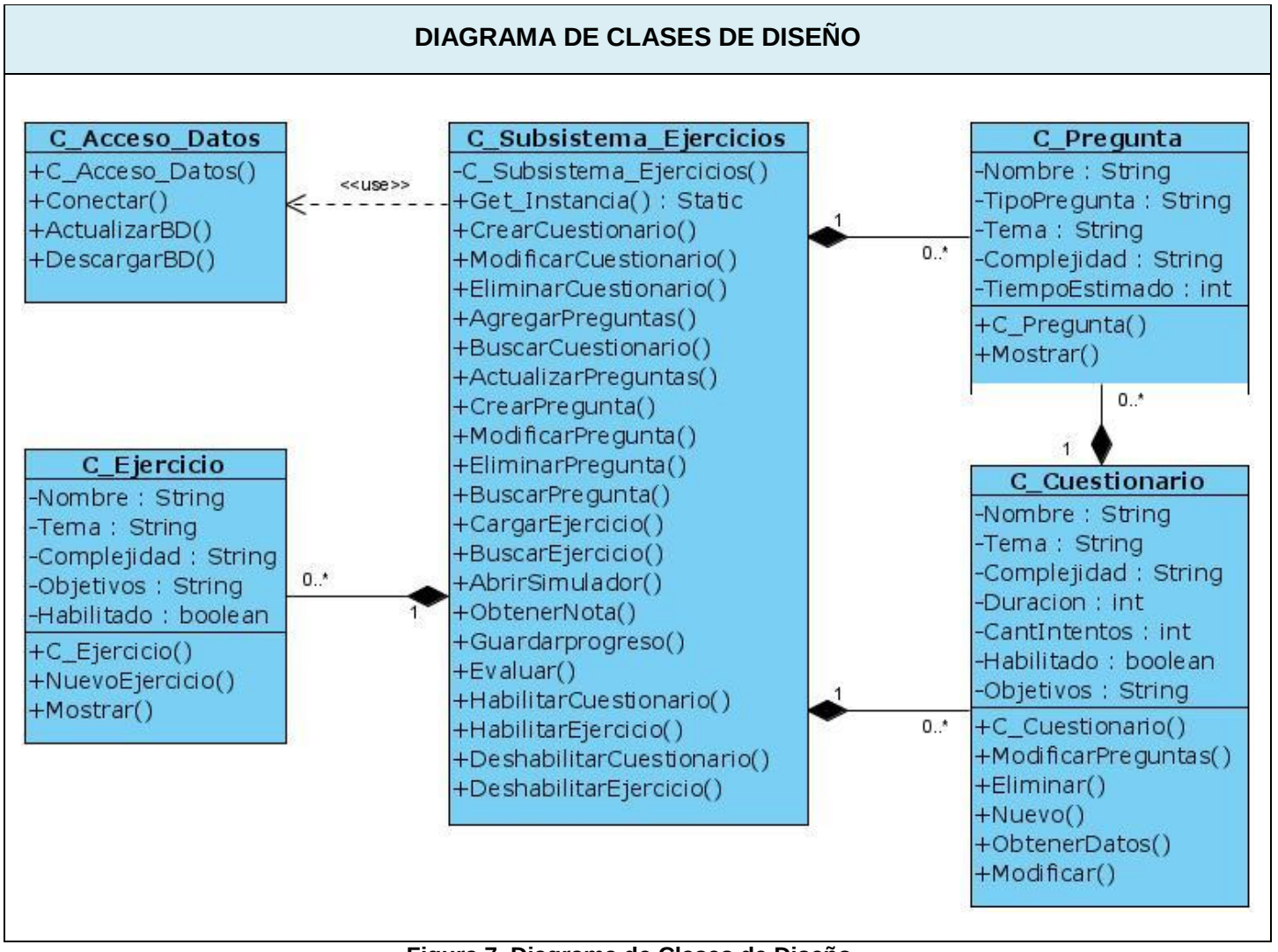


Figura 7. Diagrama de Clases de Diseño.

2.5.3.1 Patrones de diseño aplicados

Un patrón de diseño es una solución aplicable a diferentes problemas de diseño en circunstancias desiguales. Existen varios patrones popularmente conocidos entre los que destacan los GoF (acrónimo de ***G**ang **o**f **F**our, traducido en Banda de los cuatro*) y los **GRASP** (acrónimo de *General Responsibility Assignment Software Patterns,* traducido en Patrones Generales de Software para Asignación de Responsabilidades). Los primeros antes mencionados son recogidos en el libro "*Design Patterns"* y se clasifican en creacionales, estructurales y de comportamiento. Los segundos, más que patrones, se consideran una serie de buenas prácticas de aplicación, recomendables en el diseño de software.

Los patrones utilizados para el diseño del subsistema fueron los siguientes:

Patrones GoF:

- **Instancia única:** Patrón de tipo creacional que garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Este patrón se puede evidenciar en la clase controladora C_Subsistema_Ejercicios que es la única que puede crear una y sólo una instancia de sí misma.

Patrones GRASP:

- **Alta cohesión y bajo acoplamiento**

Se considera alta cohesión cuando la información que se almacena en una clase es lo más coherente y está, en la medida de lo posible, relacionada con ella. Por otra parte el bajo acoplamiento es la idea de tener las clases lo menos ligadas entre sí posible, de tal forma que al producirse una modificación en alguna de ellas, repercuta lo menos posible en el resto, potenciando así la reutilización y la independencia entre clases.

En general el DCD presenta una alta cohesión y un bajo acoplamiento. Ejemplo de ello lo constituye la clase C_Ejercicio que sólo maneja la información referente a los datos del ejercicio y mantiene relación únicamente con la clase C_Subsistema_Ejercicios, encargada de manejar los mismos, lo cual evitaría afectaciones en otras clases, en caso de producirse un error en el subsistema.

Conclusiones

Este capítulo tuvo como propósito modelar la propuesta de solución, obteniendo y transformando los requisitos en el diseño del subsistema. Los artefactos generados serán la base para su posterior implementación, fase en la cual deberán ser codificadas cada una de las funcionalidades aquí descritas, convirtiéndolo finalmente en el producto que dará solución al problema que se plantea al comienzo de este trabajo.

Capítulo 3. Validación de la solución propuesta

Introducción

En este capítulo se realiza una evaluación de la calidad de la especificación de requisitos y el diagrama de casos de uso del sistema, artefactos que forman parte de la propuesta de solución y que representan elementos fundamentales en el desarrollo de cualquier sistema. Para llevar a cabo esta verificación se utilizaron técnicas de validación de la ingeniería de requisitos y métricas para medir la calidad del software.

3.1 Validación de los requisitos

Los requisitos, una vez definidos, necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de estos describe realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos.

3.1.1 Prototipos de interfaz no funcional

Los prototipos de interfaz no funcional son muy usados para validar los requisitos del software. Representan una buena opción para mostrarle al cliente una primera versión del producto, sin funcionalidades implementadas, para que detecte y corrija omisiones o malas interpretaciones de los mismos, e incorpore nuevas ideas.

Los prototipos fueron anexados a la descripción de los CUS, con el objetivo de hacerla más explícita. Una vez aplicada la técnica se obtuvo una buena aceptación por parte del cliente, quien aprobó que en las muestras no se omite ni mal interpreta ningún requisito, satisfaciendo sus necesidades.

3.1.2 Matriz de trazabilidad

Esta técnica de validación de la IR permite verificar que cada requisito de software esté asociado con al menos un caso de uso del sistema. Para ello se hace corresponder en una tabla requisitos y casos de uso y se indica con una cruz el punto donde ambos convergen.

	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8
R1	X							
R2		X						
R3		X						
R4		X						
R5			X					
R6			X					
R7			X					
R8				X				
R9				X				
R10					X			
R11					X			
R12						X		
R13							X	
R14								X

Tabla 4. Matriz de Trazabilidad de Requisitos.

Como resultado se obtuvo que cada requisito está presente en al menos un caso de uso lo cual demuestra que el DCUS recoge todas las necesidades identificadas.

3.1.3 Métrica de la calidad de la especificación

Las métricas se aplican para valorar la calidad de los productos de ingeniería o los sistemas que se construyen, comprobar su efectividad y correspondencia de los artefactos generados atendiendo a las exigencias del cliente. Proporcionan una manera sistemática de valorar la calidad basándose en un conjunto de reglas claramente definidas y se aplican durante todo el ciclo de vida, permitiendo descubrir y corregir problemas potenciales.

Para medir la calidad de la Especificación de Requisitos de Software, se propone una lista de características entre las que predominan: especificidad (ausencia de ambigüedad), estabilidad, compleción, corrección, comprensión, capacidad de verificación, consistencia interna y externa. De ellas fueron evaluadas para este caso la especificidad y la estabilidad.

A continuación se describe el procedimiento:
Para calcular el número total de requisitos que conforman la especificación se aplica la fórmula:

NR = NF + NNF
NR = 14 + 13
NR = 27

Dónde:

NR: número de requisitos.
NF: número de requisitos funcionales.
NNF: número de requisitos no funcionales.

Para determinar la ***especificidad*** (ausencia de ambigüedad) de los requisitos se sugiere una métrica basada en la consistencia de la interpretación de los revisores para cada requisito. Cuanto más se aproxime el valor resultante a 1, menor será la ambigüedad de estos.
Se calcula Q1:

Q1 = Rui / Rt
Q1 = 14 / 14

Q1 = 1

Dónde:
Rui: Número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.
Q1: Ausencia de ambigüedad.

El cálculo arrojó un resultado óptimo, que demuestra la consistencia de los requisitos y a su vez la inexistencia de ambigüedad en la especificación.

La ***estabilidad*** es determinada con la fórmula $E = \frac{Rt - Rm}{Rt}$, donde R_m son los requisitos modificados, que es equivalente al número de requisitos. Se considera como valor óptimo para esta métrica el valor más próximo a 1.

Se sustituyen los valores:

E = (Rt – Rm) / Rt
E = (14 – 1) / 14
E = 0,93

El resultado obtenido indica que los requisitos son estables, ya que 0,93 se considera un valor de estabilidad alto, basado en el siguiente rango:

Alta (0.90 <= **E** <= 1).
Media (0.80 <= **E** < 0.90).
Baja (0.7 <= **E** < 0.80).

3.2 Modelo de métricas Orientada a Objetos para el DCUS

Para evaluar la calidad de la funcionalidad del DCUS, el Modelo de Métricas Orientada a Objetos define cuatro atributos fundamentales:

- Completitud: Grado en el que se han detallado los casos de uso más relevantes.

- Consistencia: Grado en que los casos de uso del sistema describen las interacciones entre el sistema y los usuarios.
- Correctitud: Grado en que las interacciones entre los actores y el sistema soportan adecuadamente el modelo de negocio.
- Complejidad: Grado de claridad en la presentación de los elementos que describen la claridad y el contexto del sistema.

Para clasificar los resultados obtenidos se establecen las siguientes reglas:

Alto (90% <= **E** <= 100%).

Medio (0.80 <= **E** < 0.90).

Bajo (0.7 <= **E** < 0.80).

En una primera iteración la métrica es aplicada arrojando los siguientes resultados:

Factores de Completitud	**Métricas Asociadas**	**Evaluación para el subsistema de Ejercicios del Laboratorio Virtual de Sistemas Operativos**
Factor 1. ¿Han sido definidos todos los roles relevantes de usuario encargados de generar/modificar o consultar información?	**Métrica 1.** Número de roles relevantes omitidos. Umbral < 10%	**Total de roles relevantes**: 2 Número de roles omitidos: 0
Factor 2. ¿Están definidos todos los requisitos que justifican la funcionalidad del caso de uso?	**Métrica 2.** Número de requisitos omitidos por caso de uso. Umbral < 10% **Métrica3.** Número de casos de uso que tienen requisitos omitidos. Umbral: 10%	**Total de requisitos:** 14 Número de requisitos omitidos por casos de uso: 0 **Total de casos de uso:** 8 Número de casos de uso que tienen requisitos omitidos: 0 Representa: 0%
Factor 3.	**Métrica 4.**	**Total de casos de uso:** 8

¿Se describen las condiciones de excepción relevantes que debe contemplar cada flujo de eventos?	Número de casos de uso que no describen condiciones de excepción relevante. Umbral: 20%	Número de casos de uso que no describen condiciones de excepción relevantes**:** 5 Representa: 62,5%
Factores de Consistencia	**Métricas asociadas**	**Evaluación para el subsistema de Ejercicios del Laboratorio Virtual de Sistemas Operativos**
Factor 4. ¿El nombre dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario?	**Métrica 5.** Número de casos de uso que tienen nombre incorrecto Umbral: 20%	**Total de casos de uso:** 8 Número de casos de uso que tiene nombre incorrecto: 1 Representa: 12,5%
Factor 5. ¿Representa el caso de uso una interacción observable por el actor?	**Métrica 6.** Número de casos de uso que no representan una interacción observable por un actor. Umbral: 5%	**Total de casos de uso:** 8 Número de casos de uso que no representan una interacción observable por un actor: 1 Representa: 12,5%
Factor 6. ¿La descripción del flujo de eventos se inicia con la descripción de una acción externa originada por un actor o una condición interna del sistema claramente identificable?	**Métrica 7.** Número de casos de uso cuya descripción extendida no inicia con una acción externa o con una condición monitoreada por el sistema. Umbral < 10%	**Total de casos de Uso:** 8 Número de casos de uso cuya descripción extendida no inicia con una acción externa o con una condición monitoreada por el sistema: 0 Representa: 0%
Factor 7. Si en el caso de uso intervienen más de un actor, ¿existe claridad en cuál de ellos es el actor iniciador?	**Métrica 8.** Número de casos de uso con más de un actor, que no describe cuál es el actor iniciador.	**Total de casos de Uso:** 8 Número de casos de uso con más de un actor, que no describe cuál es el actor iniciador: 0 Representa: 0%

	Umbral < 20%	
Factores de Correctitud	**Métricas Asociadas**	**Evaluación para el subsistema de Ejercicios del Laboratorio Virtual de Sistemas Operativos**
Factor 8. ¿Representa el caso de uso requisitos comprensibles por el usuario?	**Métrica 9.** Grado en que los requisitos representados por el caso de uso son comprensibles por el usuario.	**Total de requisitos:** 14 Cantidad de requisitos que no son comprensibles por el usuario: 0 Representa: 0%
	Métrica 10. Número de casos de uso en que los requisitos representados no son comprensibles por el usuario. Umbral < 5%	**Total de casos de Uso:** 8 Número de casos de uso en que los requisitos representados no son comprensibles por el usuario: 1 Representa: 12,5%
Factores de Complejidad	**Métricas Asociadas**	**Evaluación para el subsistema de Ejercicios del Laboratorio Virtual de Sistemas Operativos**
Factor 9. ¿Los elementos dentro del diagrama están adecuadamente ubicados de manera que facilitan su interpretación?	**Métrica 11.** Número de elementos del diagrama que requieren reubicación. Umbral < 30%	**Total de casos de Uso:** 8 Número de elementos del diagrama que requieren reubicación: 0 Representa: 0%

Tabla 5. Modelo de Métricas Orientada a Objeto aplicadas al Diagrama de Casos de Uso del Sistema.

El resultado obtenido demuestra un bajo grado de funcionalidad del DCUS con un valor de 81,2%, para una contribución independiente por atributo a la calidad total de 90,6% en Completitud, 78,1% de Consistencia, 56,25% de Correctitud y 100% de Complejidad.

Para verificar que hayan sido corregidas las no conformidades detectadas en la iteración anterior se realiza una segunda iteración (Ver Anexo 4) de la evaluación. Los resultados adquiridos reflejan una mejora en la calidad del artefacto con un valor de 92,2 % de funcionalidad del DCUS, consecuencia de un 90,6 % de Completitud, 78,1% de Consistencia, 100% de Correctitud y 100% de Complejidad. Teniendo en cuenta las reglas definidas para clasificar los resultados se concluye que el DCUS posee un alto grado de funcionalidad.

Los resultados de ambas iteraciones son representados en una tabla, (Ver anexo 5), donde se exponen con mayor claridad.

3.3 Métricas para validar el diseño.

Para medir la calidad del diseño se utilizaron métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto referenciadas por Pressman [16], teniendo en cuenta que este estudio brinda un esquema sencillo de implementar y que a la vez cubre los principales atributos de calidad de software.

3.3.1 Métrica tamaño operacional de clase

El Tamaño Operacional de Clase (TOC) está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

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

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

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

Tabla 6. Criterios de evaluación para la métrica TOC.

Al aplicar la métrica se obtienen los siguientes resultados:

Total de clases: 5

Total de procedimientos: 37

Promedio de procedimientos: 7,4

Cantidad de procedimientos por clases:

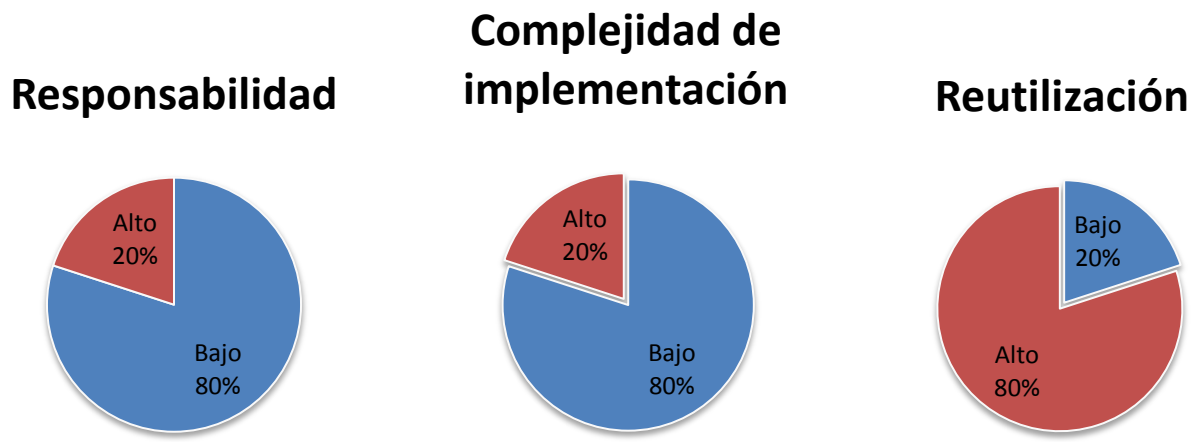
Criterios	Cantidad de clases	Porciento
Entre 1 y 7 procedimientos	4	80
Entre 8 y 13 procedimientos	0	0
Entre 14 y 19 procedimientos	0	0
Entre 20 y 25 procedimientos	1	20
Total	**5**	**100**

Tabla 7. Cantidad de procedimientos por clase.

A continuación se muestra la incidencia de los resultados de la evaluación de la métrica TOC en cada uno de los atributos:

Atributo	Categoría	Cantidad de clases	Porciento
Responsabilidad	Baja	4	80
	Media	0	0
	Alta	1	20
Complejidad de implementación	Baja	4	80
	Media	0	0
	Alta	1	20
Reutilización	Baja	1	20
	Media	0	0
	Alta	4	80

Tabla 8. Incidencia de los resultados de la evaluación de la métrica TOC por atributo.



Los resultado obtenidos demuestran que el diseño propuesto se encuentra entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (80%) posee menos cantidad de operaciones que la media registrada en las mediciones, esto fomenta la reutilización y la complejidad de implementación.

3.3.2 Métrica de relaciones entre clases

Las Relaciones entre Clases (RC) están dadas por el número de relaciones de uso entre las clases y evalúa los siguientes atributos de calidad:

- **Responsabilidad:** Un aumento del RC implica un aumento de la responsabilidad asignada a la clase.
- **Complejidad de mantenimiento:** Un aumento del RC implica un aumento en la complejidad del mantenimiento de la clase.
- **Reutilización:** Un aumento del RC implica una disminución del grado de reutilización de la clase.
- **Cantidad de pruebas:** Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

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

	Media	Entre Promedio y 2 * Promedio
	Alta	> 2 * Promedio

Tabla 9. Criterios de evaluación de la métrica RC.

Al aplicar la métrica se obtienen los siguientes resultados:

Total de clases: 5

Total de asociaciones de uso: 5

Promedio de asociaciones de uso: 1

Cantidad de dependencias por clase:

Criterio	Categoría	Cantidad de clases	Porciento
0 dependencias	Muy Bueno	0	0
1 dependencias	Bueno	2	40
2 dependencias	Regular	2	40
3 dependencias	Malo	0	0
>3 dependencias	Muy Malo	1	20
Total		5	100

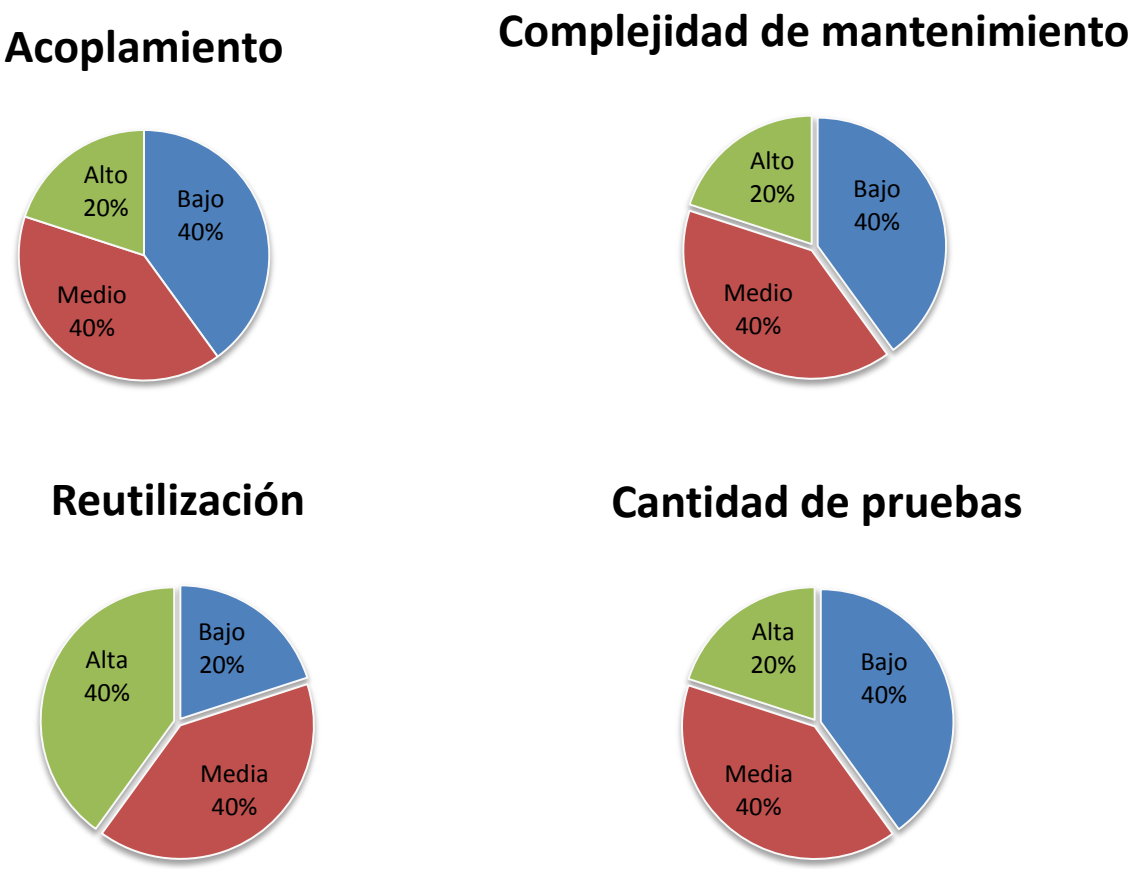
Tabla 10. Cantidad de dependencias por clasificación.

A continuación se muestra la incidencia de los resultados de la evaluación de la métrica RC en cada uno de los atributos:

Atributo	Categoría	Cantidad de clases	Porciento
Acoplamiento	Ninguno	0	0
	Bajo	2	40
	Medio	2	40
	Alto	1	20
Complejidad de	Baja	2	20

mantenimiento	Media	2	40
	Alta	1	20
Reutilización	Baja	1	20
	Media	2	40
	Alta	2	40
Cantidad de pruebas	Baja	2	40
	Media	2	40
	Alta	1	20

Tabla 11. Incidencia de los resultados de la evaluación de la métrica RC por atributo.



Los resultados obtenidos están entre los límites aceptables de calidad. Los atributos de calidad se encuentran en un nivel satisfactorio, el grado de acoplamiento es relativamente bajo, la complejidad de mantenimiento, la cantidad de pruebas y la reutilización se comportan favorablemente.

Conclusiones

Luego de aplicar las técnicas y métricas de calidad para validar los artefactos que componen la propuesta de solución y valorar los resultados se concluye:

- Los requisitos se corresponden con lo que inicialmente se pretendía.
- Los casos de uso que conforman el DCUS recogen todos los requisitos funcionales obtenidos en la etapa de captura, y los resultados de la validación demuestran que están aptos para regir el diseño, la construcción y las pruebas del subsistema.
- Las métricas aplicadas al diseño arrojaron resultados aceptables entre los límites de la calidad, lo que da fe de la calidad de los artefactos obtenidos.

Conclusiones

La necesidad de contar con ejercicios y cuestionarios que permitan a los estudiantes comprobar su autoaprendizaje dentro del Laboratorio Virtual de Sistemas Operativos, y a los profesores controlar el progreso de sus estudiantes y la calidad del proceso enseñanza - aprendizaje, da origen a la investigación que finaliza con las siguientes conclusiones:

- El estudio de las funcionalidades más representativas de los sistemas de enseñanza – aprendizaje permitió arribar a una aproximación de la solución deseada.

- Las actividades y técnicas de la Ingeniería de Requisitos utilizadas facilitaron la obtención, especificación y validación de los requisitos del subsistema, permitiendo verificar que las funcionalidades identificadas respondían a las necesidades del cliente.

- La metodología de desarrollo, el lenguaje y la herramienta de modelado seleccionada a partir del estudio realizado, facilitaron la creación de los modelos de análisis y diseño del Subsistema de Ejercicios para el Laboratorio Virtual de Sistema Operativos.

- La validación de los artefactos generados y los resultados alcanzados demostraron que es viable implementar el subsistema a partir de la solución que se proponen en este trabajo.

Recomendaciones

- Continuar con los flujos de trabajo propuestos por RUP para el desarrollo del software de manera que se obtenga una solución informática que cumpla con los requisitos y expectativas del cliente.

- Garantizar la gestión de los requisitos en las etapas posteriores para controlar los cambios que puedan surgir en los requisitos del sistema y gestionar eficientemente los riesgos que puedan atentar contra la calidad del producto.

Referencias

[1] Á. Salaverría, L. F. Ferreira, J. Martínez, J.G. Dacosta y E. Mandado. *Laboratorio virtual para el autoaprendizaje de la Electrónica Aplicada*. [En línea]. Tecnologías Aplicadas a la Enseñanza de la Electrónica, 2006. [Consultado en: marzo, 2011]. Disponible en: http://taee.euitt.upm.es/Congresosv2/2006/papers/2006SD108.pdf

[2] Gutiérrez Atoche, Egberto Serafín. *Laboratorio Virtual de Física en el área de circuitos de Corriente Continua y Corriente Alterna.* [En línea]. [Consultado en: marzo, 2011]. Disponible en: http://www.monografias.com/trabajos46/laboratorio-virtual-fisica/laboratorio-virtual-fisica.shtml

[3] Spínola Elías, Elías. *Internet en el aula.* [En línea]. Maestría en Tecnología Educativa del Centro Universitario del Oriente de Hidalgo. [Consultado en: marzo, 2011]. Disponible en: http://webblogenladocenciauniversitaria.blogia.com/2010/091001-web-blog-en-la-docencia-universitaria.php

[4] L. Rosado, J. R. Herreros. *Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la física.* [En línea]. Recent Research Developments in Learning Technologies, 2005. [Consultado en: marzo, 2011]. Disponible en: http://www.uv.es/eees/archivo/286.pdf

[5] Real Academia Española. *Diccionario de la lengua española*. 22 ed. 2001. [Consultado en: mayo, 2011]. Disponible en: http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=Evaluaci%C3%B3n

[6] Catari Padilla, Gabriel. *La autoevaluación como estrategia de aprendizaje para atender a la diversidad*. [En línea]. [Consultado en: mayo, 2011]. Disponible en: http://www.monografias.com/trabajos74/autoevaluacion-estrategia-aprendizaje-diversidad/autoevaluacion-estrategia-aprendizaje-diversidad.shtml

[7] Ángel García-Beltrán, Raquel Martínez, José-Alberto Jaén, Santiago Tapia. *La autoevaluación como actividad docente en entornos virtuales de aprendizaje/enseñanza.* [En línea]. [Consultado en: mayo, 2011]. Disponible en: http://www.um.es/ead/red/M6/garcia_beltran.pdf

[8] Ingeniería de Requerimientos. Ingeniería De Software. [En línea]. Scribd. [Consultado en: marzo, 2011]. Disponible en: http://es.scribd.com/doc/51379254/Ingenieria-De-Requerimientos

[9] Jacobson, Ivar, Booch, G y Rumbaugh, J. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid: The Addison- Wesley Object Technology Series, 2000.

[10] Solís, Manuel Calero. *Una explicación de la programación extrema (XP)*. [En línea]. 2003. [Consultado en: marzo, 2011]. Disponible en: http://www.willydev.net/descargas/prev/ExplicaXp.pdf.

[11] *¿Qué es SCRUM?* [En línea]. proyectosagiles.org, 2010. [Consultado en: marzo, 2011]. Disponible en: http://www.proyectosagiles.org/que-es-scrum.

[12] Gracia, A. *Análisis y diseño de un sistema automatizado para el control de los recursos humanos en los polos productivos de la facultad 9*. Universidad de las Ciencias Informáticas. 2009.

[13] Hernández, Berta. *Informe del rol de arquitecto del subsistema de modelos y propiedades de un simulador para la industria química*. Universidad de las Ciencias Informáticas. 2009.

[14] Mora, Beatriz FR, García Félix, Piattini Mario. *Experiencia en transformación de modelos de procesos de negocios desde BPMN a XPDL*. [En línea]. [Consultado en: marzo, 2011]. Disponible en: http://kuainasi.ciens.ucv.ve/ideas07/documentos/articulos_ideas/Articulo35.pdf

[15] *Visual Paradigm for UML*. [En línea]. 2007. [Consultado en: marzo, 2011]. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(MÍ)_14720_p/

[16] Roger S. Pressman. Ingeniería de software un enfoque práctico. Quinta edición.1998.

Bibliografía

1. Catari Padilla, Gabriel. *La autoevaluación como estrategia de aprendizaje para atender a la diversidad*. [En línea]. [Consultado en: marzo, 2011]. Disponible en: http://www.monografias.com/trabajos74/autoevaluacion-estrategia-aprendizaje-diversidad/autoevaluacion-estrategia-aprendizaje-diversidad.shtml

2. Ángel García-Beltrán, Raquel Martínez, José-Alberto Jaén, Santiago Tapia. *La autoevaluación como actividad docente en entornos virtuales de aprendizaje/enseñanza.* [En línea]. [Consultado en: marzo, 2011]. Disponible en: http://www.um.es/ead/red/M6/garcia_beltran.pdf

3. García-Beltrán, Ángel. Martínez, Raquel. *AulaWeb: un sistema para la gestión, evaluación y seguimiento de asignaturas.* [En línea]. [Consultado en: marzo, 2011]. Disponible en: http://www.dii.etsii.upm.es/documents/AulaWebIndustriaConFiguras.pdf

4. Escalona, M J y Koch, Nora. *Ingeniería de Requisitos en Aplicaciones para la Web, un estudio comparativo*. 2004.

5. Pérez González, Dairelis. Hernández Pérez, Grenia. *Guía para realizar Ingeniería de Requisitos a la línea de productos informáticos que utilizan tecnología multimedia*. Universidad de las Ciencias Informáticas. 2010.

6. CRAFTWARE. *Prototipos de la Interfaz Usuaria con Axure RP Pro.* Santiago de Chile. [En línea]. [Consultado en: mayo, 2011] Disponible en: http://www.hotfrog.cl/Products/Axure-RP.

7. Canos, José H. y Patricio Letelier. *Metodologías Ágiles en el Desarrollo de Software*. Valencia: s.n., 2003.

8. Mendoza, María y Sánchez. *Metodologías De Desarrollo De Software*. Perú: s.n., 2004.

9. Pérez Pereira, Tamara. Cruz Álvarez, Yailén. *Propuesta de un Laboratorio Virtual de Física I sobre Fuerza de Fricción Seca. Estudio Preliminar*. Universidad de las Ciencias Informáticas. 2007.

Glosario de abreviaturas

TIC - Tecnologías de la Información y la Comunicación.

EVA - Entorno Virtual de Aprendizaje.

UCI - Universidad de las Ciencias Informáticas.

IR - Ingeniería de Requisitos.

RUP (*Rational Unify Process*) - Proceso Unificado de Rational.

CU - Caso de Uso.

XP (*Extreme Programming*) - Programación Extrema.

UML (*Unify Model Language*) - Lenguaje Unificado de Modelado.

BPMN (*Business Process Modeling Notation*) - Notación de Modelado de Procesos de Negocio.

CASE (*Computer Aided Software Engineering*) - Ingeniería de Software Asistida por Computadora.

RF - Requisito Funcional.

RNF - Requisito No Funcional.

DCUS - Diagrama de Casos de Uso del Sistema.

CRUD (*Creating, Reading, Updating, Deleting*) - Crear, Leer, Actualizar, Eliminar.

CUS - Casos de Uso del Sistema.

DCA - Diagrama de Clases de Análisis.

GoF (*Gang of Four)* - Grupo de los cuatro.

GRASP (*General Responsibility Assignment Software Patterns*) - Patrones Generales de Software para Asignación de Responsabilidades.

NR - Número de Requisitos.

NF - Número de requisitos Funcionales.

NNF - Número de requisitos No Funcionales.

Glosario de términos

Aprendizaje: Es un proceso en el que participa activamente el alumno, dirigido por el docente, apropiándose el primero de conocimientos, habilidades y capacidades, en comunicación con los otros, en un proceso de socialización que favorece la formación de valores, es la actividad que favorece la asimilación de un proceso, la enseñanza.

Aprendizaje colaborativo: Es un método de instrucción en el cual los alumnos trabajan en pequeños equipos hacia una meta en común: aprender.

Artefacto: Producto tangible del proyecto que es producido, modificado y usado por las actividades.

Autoaprendizaje: Es la capacidad del sujeto de hacerse responsable de su propio proceso de aprendizaje, de ser autogestivo y tener una curiosidad intelectual, así como el goce que aporta el desarrollo intelectual.

Autoevaluación: La Tecnología Educativa actual y la que se prevé en el futuro se preocupa tanto por el control que la sociedad debe ejercer por la calidad de la educación, como por una nueva forma de control vinculada al aprendizaje del alumno, este es el llamado “control introyectivo” o “comunicación intrapersonal”, la autoevaluación de su aprendizaje, que se manifiesta en fenómenos de conciencia y autodeterminación del estudiante, principal rector del proceso.

Clase: Parte de una jerarquía representada por una línea que va desde cualquier punto del cuadro jerárquico de clasificación a todos los expedientes que quedan por debajo.

Cliente: Individuo que abre espacios y compromete recursos (de tiempo, económicos, de identidad) para interactuar con otro individuo. Es quien recibe beneficios de exigencias impuestas.

Enseñanza - aprendizaje: Constituye la vía mediatizadora esencial para la apropiación de conocimientos, habilidades, hábitos, normas de relación, de comportamiento y valores, legados por la humanidad, que se expresan en el contenido de enseñanza, en estrecho vínculo con el resto de las actividades docentes y extradocentes que realizan los estudiantes.

Entornos simulados: Se entiende por simulador la representación de un escenario que imita a la realidad haciendo posible la reproducción de lo cotidiano. Hay experiencias de la vida real que serían muy costas llevarlas a la práctica. Los simuladores virtuales de aprendizaje favorecen la adquisición del saber hacer, que es sin dudas lo más difícil de adquirir en el mundo real.

Entorno Virtual de Aprendizaje: Espacio con accesos restringidos, concebido y diseñado para que las personas que acceden a él desarrollen procesos de incorporación de habilidades y saberes, mediante sistemas telemáticos.

Evaluación: Es el proceso para comprobar y valorar el cumplimiento de los objetivos propuestos y la dirección didáctica de la enseñanza y el aprendizaje en sus momentos de orientación y ejecución. Se deberán propiciar actividades que estimulen la autoevaluación por los estudiantes, así como las acciones de control y valoración del trabajo de los otros.

Herramientas: Programas, subprogramas e instrucciones para facilitar la puesta en marcha de otros programas o acciones.

Modelado: Modelar es desarrollar una descripción lo más exacta posible de un sistema y de las actividades llevadas a cabo en él.

Moodle: Objeto de aprendizaje dinámico orientado a objetos y modular (Modular Object-Oriented Dynamic Learning Environment). Se define como un sistema de gestión de cursos, un paquete de software diseñado para ayudar al profesor o asesor a crear fácilmente cursos de calidad en línea, se expresa en diferentes actividades o módulos tales como: Foros, Diarios, Cuestionarios, Tareas y otros, que facilitan el aprendizaje desde una posición participativa.

Negocio: Ambiente o entorno en cual está enmarcado el problema.

Servidor: Una aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes. Algunos servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de un ordenador y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final. Es posible que un ordenador cumpla simultáneamente las funciones de cliente y de servidor.

Servidor Web: Programa que implementa el protocolo HTTP (Hypertext Transfer Protocol). Está diseñado para transferir lo que llamamos hipertextos, páginas Web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

Simulación: Recreación de procesos que se dan en la realidad mediante la construcción de modelos que resultan del desarrollo de ciertas aplicaciones específicas. Los programas de simulación están muy extendidos y tienen capacidades variadas, desde sencillos juegos de ordenador hasta potentes aplicaciones que permiten la experimentación industrial.

Software: Conjunto de instrucciones escritas en un determinado lenguaje, que dirigen a un ordenador para la ejecución de una serie de operaciones, con el objetivo de resolver un problema que se ha definido previamente.

Tecnologías de la Información y la Comunicación: Agrupan los elementos y las técnicas utilizadas en el tratamiento y la transmisión de las informaciones, principalmente de informática, internet y telecomunicaciones.

Virtual: Ser en esencia o efecto, no en realidad. Algo que tiene sentido en el contexto de una simulación, donde toma aspecto de existencia.