“Universidad de las Ciencias Informáticas”

UCI

Análisis y Diseño del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

Tesis de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

Autores: Danirys Ortega González.

Reinaldo Rodríguez Veitia.

Tutor: M.Sc. Yoan Martínez Márquez.

Ing. Reinier Castillo González.

Co-Tutor: Lic. Moisés A. Mayet Solano.

La Habana, Cuba.
2010

"Para pensar hay que existir, para dialogar hay que sobrevivir y para sobrevivir hay que luchar."

Fidel Castro

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Danirys Ortega González

_____

Firma del Autor

Reinaldo Rodríguez Veitia

_____

Firma del Autor

Yoan Martínez Márquez

_____

Firma del Tutor

Agradecimientos

A lo más grande que he tenido en el mundo, por su incondicional apoyo, preocupación y por todos los años de sacrificio, gracias por tu dedicación y enseñanza. Aunque ya no estés a mi lado siempre estarás en mi corazón. Te extraño mucho. TE QUIERO CON TODA MI ALMA MAMITA.

A mi papá Andrés por el apoyo incondicional que recibo de ti todos los días, por educarme y guiarme por el camino correcto, por todos los sacrificios que has dedicado a mí. Siempre estaré orgullosa de ti.

A mi hermana Damarys y a mi gordito Anthony que con todo su cariño han sabido animarme y transmitirme seguridad y fuerzas para salir adelante.

A mi abuela Rosa por brindarme sus sabios consejos y experiencias y por dedicar todo su tiempo a preocuparse por mí. ERES MI ABUELITA LINDA.

A una persona que llegó a mi vida y sigue en ella y seguirá porque lo quiero mucho, mi novio y compañero de tesis Reinaldo por compartir a mi lado estos 5 inolvidables años, por recordarme a cada momento la felicidad que hemos pasado juntos, por entenderme y apoyarme cuando más lo necesitaba, por aceptarme con mis defectos y virtudes, por hacerme sentir parte de ti, quiero que sepas que te AMO MUCHO MI PACHI.

A mis Tutores por todo su ejemplo y dedicación. Por la ayuda que nos brindaron durante esta última etapa, que nos sirvió de guía en todo momento. A ustedes MUCHAS GRACIAS.

A todas las personas maravillosas que he tenido la suerte de conocer en estos años, a mis amigas de la infancia Pucha, Mariem y Samay. Sin dejar atrás a mis amigas de la universidad Dunia, Anisley, Bianka, Claudia, Maidolis, Yami, Anyelin, Dalgys, Yuli, Lisy y Lianni, junto a ellas pasé los mejores años de la universidad.

A todos muchas gracias. *Danirys*

Agradecimientos

A mi mamá por todo su amor, comprensión y cariño durante toda mi vida, por confiar siempre en mí.

A mis abuelos por guiarme y apoyarme durante todo este tiempo y confiar en mí.

A mi papá por apoyarme durante todos estos años.

A esa persona especial en mi vida que ha sabido alumbrar mi camino durante estos años, la que siempre estaba cuando más la necesitaba, a mi novia y amiga Danirys.

A mis tíos por apoyarme y saber aconsejarme durante todos estos años de estudio.

A mis primos y primas, por apoyarme y ayudarme durante este tiempo.

A mis tutores por saber guiarnos en la elaboración de este trabajo.

A mis amigos por apoyarme y brindarme su amistad sin condición.

A todos mis compañeros de aula y de proyecto que compartieron conmigo durante todos estos años.

A todos en general.

Reinaldo

Dedicatoria

A mi amada mamá que más que por mí siento que todo lo hago por ella y para ella, porque se lo merece aunque ya no esté a mi lado.

A mi papá que tanto ha hecho por mí y por mis sueños.

A mis padres les dedico todos mis éxitos. A ellos les debo todo lo que soy.

A mi segunda mamá, mi abuelita linda, que tanto quiero. Por siempre apoyarme y estar conmigo a cada momento.

A mi hermana por indicarme siempre el camino correcto.

A mi adorado sobrinito, mi gordito chiquitico, que vino al mundo para brindarle mucho amor y alegría a nuestra familia.

A mi novio, mi gran amor, por estar ahí cuando más lo necesité, por el inmenso apoyo que siempre me has dado, por todos tus consejos, por ser mi inspiración para todo, por enseñarme tantas cosas, sabes que nunca te olvidaré. A ti mi Rey. Ojalá la vida me permita estar siempre a mi lado.

A mi familia que de una forma u otra siempre me han tendido la mano, a ustedes también va esta dedicatoria.

A mis amigos que estuvieron conmigo en los momentos más felices y difíciles de mi vida.

Y al compañero Fidel Castro por haber creado esta maravillosa universidad de la cual yo he formado parte y me siento muy orgullosa por ello.

Danirys

Dedicatoria

A mi mamá por su apoyo y comprensión durante todos estos años. Por tener un boto de confianza en mí, por todo su sacrificio y por todo el amor que siento por ti.

A mis abuelos por todo el apoyo brindado y por confiar siempre en mí.

A mi hermana por estar siempre conmigo y compartir todo su amor permitiéndome ser su guía durante este tiempo.

A mi familia y a mí papá por darme siempre su apoyo y confianza durante todos estos años.

A una persona especial que durante todos estos años ha compartido conmigo buenos y malos momentos, a él amor de mi vida Danirys. Gracias por estar aquí conmigo y por brindarme todo su amor, comprensión y apoyo siempre. Por quererme como soy y amarme sin condición.

A mis primos en especial a Jarcel por su ayuda durante el transcurso de mi carrera.

A mis tíos que de una forma u otra han sabido apoyarme y darme fuerzas para salir adelante.

A mis tutores por el apoyo brindado durante todo este tiempo.

A todos mis compañeros del proyecto que de una manera u otra han contribuido a la realización de este trabajo.

A todos mis compañeros y amigos que han compartido conmigo durante este tiempo.

… a todos los que de una forma u otra han contribuido con mi formación como profesional y como persona.

A todos ustedes.

Muchas Gracias. *Reinaldo*

RESUMEN

En la actualidad las Tecnologías de la Información y las Comunicaciones han tomado un gigantesco auge y la esfera educativa se ha ido ajustando a este proceso. En este contexto ocupa un espacio determinante el autoaprendizaje, la autogestión del conocimiento y la autonomía de los estudiantes en el proceso de aprendizaje. Esta realidad unida a la necesidad de personalizar el proceso de aprendizaje incrementa la complejidad en el desarrollo de la evaluación para el aprendizaje autónomo. Los sistemas tutores inteligentes se presentan como una propuesta novedosa dentro de las tecnologías en el proceso educativo.

La presente investigación persigue como objetivo realizar el análisis y diseño de un Sistema Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas. La misma se desarrolla en el proyecto de Innovación Pedagógica: Metodología de Evaluación para el Aprendizaje Autónomo de Idiomas.

Para ello se realizó un estudio sobre: conceptos asociados a Sistemas Tutores Inteligentes (STI), el comportamiento de las funcionalidades para el STI en otros sistemas similares en el mundo, metodologías de desarrollo de software, herramientas de modelado, lenguajes de programación, sistemas gestores de base de datos, además de algunos aspectos asociados al flujo de trabajo Análisis y Diseño definido en el Proceso Unificado de Desarrollo (RUP).

El sistema brinda a los aprendices un ambiente alternativo de evaluación para el aprendizaje autónomo de idioma. Para la construcción de esta aplicación fue necesario realizar un levantamiento de requisitos con el usuario, los cuales fueron agrupados posteriormente en casos de uso. Una vez obtenidos los casos de uso, se realizaron los diagramas de clases del análisis y de colaboración. Posteriormente se realizaron los diagramas de clases del diseño, identificando los elementos que deben ser implementados. Al finalizar con la elaboración de los artefactos del diseño, se aplicaron métricas y técnicas dirigidas a evaluar la calidad de los artefactos generados, obteniéndose resultados satisfactorios.

PALABRAS CLAVES

“Sistema Tutor Inteligente, aprendizaje, autoaprendizaje, autonomía, autogestión, Análisis, Diseño.”

ÍNDICE

INTRODUCCIÓN

En la actualidad con el continuo avance de las Tecnologías de la Información y las Comunicaciones, la esfera educativa se ha ido ajustando a este proceso y hoy se pueden probar escenarios virtuales con el fin de mejorar el proceso de enseñanza-aprendizaje.

El aprendizaje es una etapa importante en el desarrollo de toda actividad educativa, por lo que es preciso exigir para esto las más calificadas ayudas tecnológicas. Se plantea por tanto el uso de medios didácticos que sirvan de apoyo y que hagan más fácil el trabajo de profesores y alumnos.

En los primeros años de la década de los 60 surgieron las aplicaciones educativas, las cuales tenían entre sus principales enfoques la Instrucción Asistida por Computador (IAC). Más tarde, en los años setenta se empiezan a desarrollar los Sistemas Tutoriales Inteligentes (STI), partiendo de la arquitectura de tres módulos del modelo propuesto por Carbonel (Carbonel, 1970). Entre los STI desarrollados se pueden destacar el Scholar (Carbonel, 1970), Why (Stevens, 1977), el Sophie (Brown, 1982), entre otros. Los investigadores para el desarrollo de los STI han empleado técnicas de Inteligencia Artificial (IA), utilizando para esto, métodos como: redes neuronales, redes bayesianas y la minería de datos.

Un STI actúa como un tutor personal del estudiante y posee la capacidad de proceder de acuerdo a las necesidades más complejas del mismo. Con la creación de este tipo de sistemas, se puede reducir el peso de trabajo de los docentes, en ocasiones tales, que son altos los índices de alumnos respecto a los profesores en los cursos. De esta manera se personaliza el ambiente de aprendizaje, requiriendo así menos recursos humanos, conjuntamente con esto, se pueden ajustar los horarios de estudio para los estudiantes, permitiendo a los mismos interactuar con el sistema según su propio ritmo de aprendizaje.

En el entorno mundial, en la actualidad, los STI son utilizados en el aprendizaje de diversas materias y aunque se ha experimentado un creciente interés por el aprendizaje autónomo de idioma extranjero, no se tiene conocimiento de que se utilicen estos sistemas en la evaluación de este tipo de aprendizaje.

Los mayores avances en la evaluación del aprendizaje autónomo de idiomas extranjeros se concentran entonces en los exámenes internacionales estandarizados para la certificación del nivel en la Lengua Inglesa como el TOEFL (Test of English as a Foreign Language), TOEIC (Test of English for International Communication) y IELTS (International English Language Testing System). Estos exámenes no siempre

reflejan el avance del proceso de aprendizaje sino sólo los resultados finales del proceso evaluativo. Además de ser usados con fines comerciales.

Cuba se encuentra inmersa en un proceso de informatización y desarrollo informático, el cual requiere de profesionales calificados que le permitan desarrollar habilidades e intercambiar en los diferentes ámbitos sociales, para lo cual se hace necesario conocer las lenguas extranjeras para promover los lazos de amistad y cooperación entre los diferentes sectores, para ello se encuentra enfrascado en la creación de varios Centros de Auto-Aprendizaje y Servicios de Idiomas Extranjeros conocidos como (CASIE).

La Universidad de Ciencias Informáticas (UCI) es una institución que se encuentra inmersa en este proceso, por el papel que juegan en la informatización de todo el país, encontrándose las lenguas extranjeras dentro de una de sus prioridades, para ello tiene como tareas más urgentes en la actualidad la creación de métodos y sistemas que ayuden a solucionar el problema de encontrar formas óptimas y regímenes de trabajo en el aprendizaje de idioma.

Los primeros intentos por realizar un sistema automatizado para la gestión de la evaluación en la universidad fue la creación del proyecto Centro Virtual de Auto-aprendizaje de Lenguas Extranjeras (CEVALE), para la asignatura de Lenguas Extranjeras, pero el mismo fracasó por diversas razones. Existe también el Centro de Autoaprendizaje Virtual de Idiomas Extranjeros (Eva Moodle), en el que estudiantes y docentes planifican, acuerdan, desarrollan y participan activamente en experiencias orientadas al logro del aprendizaje de los estudiantes. Sin embargo el EVA Moodle no admite integrar los procesos de diagnósticos, evaluación y acreditación de los estudiantes, además no permite a los docentes integrar en una sola herramienta los procesos de encuestar, procesar y generar reportes de las encuestas realizadas a los estudiantes.

En este contexto se desarrolla el proyecto de Innovación Pedagógica: Metodología de Evaluación para Aprendizaje Autónomo de Idiomas. Se cuenta con la experiencia de los profesores de Idiomas en el desarrollo de los procesos de gestión de la evaluación para aprendizaje autónomo, con las limitaciones relacionadas previamente y se desarrolla una metodología de evaluación para el aprendizaje autónomo de idiomas sustentado en un modelo pedagógico no presencial (a distancia). Sin embargo no se cuenta con un sistema informático que determine el estilo o estilos de aprendizaje, las estrategias de aprendizaje, el perfil de inteligencia y los objetivos del usuario partiendo de un conjunto de encuestas iniciales, además

del estado actual de idioma del mismo, generando un plan de actividades evaluativas para aprendizaje autónomo a seguir durante el entrenamiento y logrando acreditar el nivel de idioma de forma autónoma. Por lo que como parte de sus resultados del proyecto, se plantea el desarrollo de un Sistema Tutor Virtual de Evaluación para Aprendizaje Autónomo de Idiomas.

Teniendo en cuenta lo anteriormente expuesto, el **problema a resolver** queda formulado de la siguiente manera: ¿Cómo traducir los procesos de gestión de evaluación para aprendizaje autónomo de idiomas extranjeros en la Universidad de las Ciencias Informáticas, a una notación entendible para los desarrolladores permitiendo su posterior implementación?

Por tanto el **objeto de estudio** corresponde a la Ingeniería de Software.

Definiendo el **campo de acción** como los Flujos de trabajo Requisitos, Análisis y Diseño definido en la metodología RUP.

Se plantea como **objetivo general** desarrollar el análisis y diseño del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

Como **idea a defender** el análisis y diseño del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas, facilitará la conversión de los requisitos del usuario en artefactos entendibles para los desarrolladores.

Objetivos específicos:

- Elaborar el marco teórico de la investigación.
- Definir los procesos de negocio y describir los requisitos funcionales del sistema.
- Diseñar la propuesta de un STI que gestione el aprendizaje autónomo de idiomas extranjeros.
- Validar los requisitos y el diseño de clases.

Para darle cumplimiento a los objetivos planteados se han trazado las siguientes **tareas de la investigación**:

- Valoración de la revisión bibliográfica sobre el aprendizaje autónomo de idiomas extranjeros y tutores inteligentes a nivel mundial y en Cuba.
- Selección de las tendencias y tecnologías actuales más apropiadas para la elaboración del sistema.
- Identificación y especificación de los requisitos que debe cumplir el sistema.
- Definición y especificación de los casos de usos.
- Realización del modelado del diseño.
- Validación de los requisitos identificados que debe cumplir el sistema.
- Validación de las clases del diseño.

Diseño Metodológico

Los **Métodos Teóricos** utilizados para la realización de este trabajo son:

Analítico-Sintético: Este método es utilizado en el análisis de la bibliografía donde se resumen los elementos más importantes de la gestión del aprendizaje teniendo en cuenta el objeto de la investigación.

Análisis Histórico Lógico: Este método es de suma importancia y es utilizado para realizar un profundo análisis de cómo ha evolucionado la gestión del aprendizaje durante los años.

Modelación: Este método es utilizado para estudiar la realidad mediante diversos modelos y diagramas que ayudarán a comprender el proceso en su totalidad. Ejemplos de esos modelos y diagramas son los diagramas de casos de usos, los diagramas de clases del análisis y el modelo de diseño.

El **Método Empírico** utilizado para la realización de este trabajo es:

Entrevista: Mediante la aplicación de este método se llevaron a cabo entrevistas no formales con el cliente donde se obtuvo la mayor cantidad de información posible, además se pudo determinar las características de la propuesta de solución teniendo en cuenta las necesidades del cliente.

CAPÍTULO 1: Fundamentación Teórica

Introducción

Como una de las propuestas de solución a la problemática planteada en la introducción de este trabajo se pensó en realizar el análisis y diseño del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas. Por tanto en este capítulo se pretende abordar los aspectos y conceptos relacionados con los STI. Se hará un análisis de la descripción actual del dominio del problema y de otras soluciones existentes en nuestro país y en el mundo, tales como: el STI ANDES, Sistema Tutorial Inteligente para el Auto entrenamiento en Sintonización de Sistemas de Control (STISIN), Autotutor y el Sistema Tutor Inteligente para el Diagnóstico y Tratamiento de Infecciones de Transmisión Sexual (STIITS). Se realizará un estudio de las metodologías de desarrollo existentes y se explicarán las herramientas escogidas para dar solución a la problemática propuesta, pretendiendo dejar sentadas las bases teóricas para un correcto análisis y diseño del software.

1.1 Conceptos asociados al dominio del problema

Aprendizaje Autónomo

Es la facultad para dirigir, controlar, regular y evaluar su forma de aprender de manera consciente e intencionada haciendo uso de Estrategias de Aprendizaje. (1)

Gestión del Aprendizaje

Es la actividad que permite generar,compartir o distribuir y utilizar el conocimiento táctico (know-how) y explícito(formal) existente en un determinado espacio, para que los individuos y las comunidades lo apliquen cuando deben enfrentar sus necesidades de desarrollo. Especificamente se pudiera decir que es la acción que una persona realiza cuando desea cumplir una meta determinada dentro del proceso de aprendizaje autónomo. (2)

Estilos de Aprendizaje

Es el término que se refiere al hecho de que cada persona utiliza su propio método o estrategias a la hora de aprender. Aunque las estrategias varían según lo que se quiere aprender, cada uno tiende a desarrollar ciertas preferencias o tendencias globales. Esas preferencias o tendencias a utilizar, más unas determinadas maneras de aprender que otras, constituyen nuestro estilo de aprendizaje. Se resumen en tres estilos de aprendizaje: estilo visual, estilo auditivo y estilo táctil kinestésico. (3)

Entorno Virtual de Aprendizaje

Es un espacio con accesos restringidos, concebido y diseñado para que las personas que acceden a él desarrollen procesos de incorporación de habilidades y saberes. Permite la creación y mantenimiento de comunidades virtuales, proporcionando los servicios con los que cada comunidad se identifica, y que garantizan la integración, enriquecimiento y fidelidad de sus usuarios. El EVA establece una red de comunicación total entre todos sus usuarios, potenciando el aprendizaje, la cooperación, la creación de nuevas iniciativas con resultados altamente positivos. (4)

Gestión de Evaluación

Es un conjunto de operaciones que se realizan 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. (5)

Sistema Tutor Inteligente

Giraffa (1997) (6) los delimita como: “un sistema que incorpora técnicas de IA (Inteligencia Artificial) a fin de crear un ambiente que considere los diversos estilos cognitivos de los alumnos que utilizan el programa”.

1.2 Descripción actual del dominio del problema

Los Centros de Autoaprendizaje y Servicios de Idiomas Extranjeros en la UCI se concibieron como un recurso importante para apoyar el desarrollo de estrategias que propicien el aprendizaje autónomo a través de las experiencias educativas de las diferentes asignaturas de las disciplinas de idiomas extranjeros y para apoyar además el desarrollo en idiomas extranjeros del claustro de profesores. Se busca que los estudiantes adquieran las competencias comunicativas y lingüísticas y que a la vez incorporen estrategias que les apoyen desarrollar una actitud de autonomía.

Actualmente en la UCI todos los usuarios (estudiantes, profesores y trabajadores) tienen acceso a los CASIE con el fin de superarse e incrementar el nivel lingüístico. Existe también el Entorno Virtual de Aprendizaje (EVA Moodle), el cual permite el estudio de manera autónoma de idiomas extranjeros. Incorporando a docentes y alumnos a un sistema que les permite evaluar los conocimientos del estudiante.

El EVA Moodle asesora en línea a la gran mayoría de los estudiantes de nuestra universidad, estos sistemas permiten no sólo la ubicación de recurso sino también las diversas formas de comunicación entre los miembros de la comunidad de aprendizaje. Se propicia el aprendizaje en grupos llamados comunidades, además del aprendizaje individual. Sin embargo no satisface todas las necesidades del modelo pedagógico utilizado en la disciplina Idiomas Extranjeros, específicamente en la gestión de evaluación para aprendizaje autónomo en los Centros de Autoaprendizaje y Servicios de Idiomas Extranjeros.

Con la creación de un Sistema Tutor Inteligente se pretende obtener un ambiente alternativo de evaluación, que les permita a los estudiantes trabajar de manera independiente, profundizando en sus debilidades y en los temas que son de su interés, además de evaluar de manera efectiva el desarrollo de las habilidades comunicativas de forma autónoma. Un STI que facilite el proceso de gestión de evaluación para aprendizaje mediante un proceso de adaptación a los estilos de aprendizaje de los estudiantes y que sea capaz de generar posibles planes de actividades evaluativas para los distintos niveles y fases del aprendizaje de lenguas extranjeras de forma personalizada. De acuerdo a su ruta de aprendizaje el usuario accede a los recursos debidamente clasificados con tales fines. En todo momento el usuario cuenta con la posibilidad de autoevaluarse y autorregular su actividad, consultando al asesor y redefiniendo su plan de evaluación para aprendizaje autónomo.

1.3 Análisis de otras soluciones existentes

Los Sistemas Tutores Inteligentes son sistemas que proporcionan aprendizaje y/o formación personalizada. Se basan en tres componentes principales:

- Conocimiento de los contenidos.
- Conocimiento del alumno.
- Conocimiento de estrategias o metodologías de aprendizaje.

A través de la interacción entre los módulos básicos, los STI son capaces de determinar lo que sabe el estudiante y cómo va en su progreso, por lo que la enseñanza, se puede ajustar según las necesidades del estudiante, sin la presencia de un tutor humano. Estos sistemas prometen transformar radicalmente nuestro concepto de aprendizaje online.

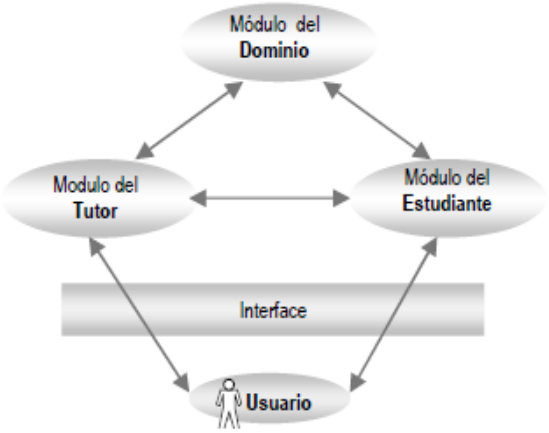


Figura 1: Interacción de los Módulos de un Sistema Tutor inteligente.

Los STI son utilizados actualmente para el autoaprendizaje de diversas materias, ayudando a los profesores a mantener una asesoría en línea con un mayor número de estudiantes. Se han implementado para ello diversos STI a nivel mundial, y en Cuba, encontrándose los siguientes:

- Sistema Tutor Inteligente **ANDES**.

El sistema Andes fue desarrollado por el equipo de Kurt Van Lehn de la Universidad de Pittsburg. El sistema se encarga de guiar a los estudiantes mientras estos resuelven problemas y ejercicios, en cursos de introducción a la Física. Cuando el estudiante pide ayuda en medio de un ejercicio, el sistema aporta pistas para avanzar en la solución o indica qué ha fallado en algún paso anterior. Andes se probó con éxito durante 5 años en la Academia Naval de los Estados Unidos y puede descargarse gratuitamente.

- Sistema Tutorial Inteligente para el Auto entrenamiento en Sintonización de Sistemas de Control (**STISIN**).

El STISIN se concibió bajo la premisa de desarrollar un sistema que se acercara lo más posible a una forma de enseñanza de acuerdo con la teoría del campo cognoscitivo. Esto significó que el estudiante pudiera interactuar con el sistema sin que éste se impusiera siempre en el proceso de enseñanza-aprendizaje. El STISIN está orientado a la instrucción de técnicas de sintonización de sistemas de control. Además es un prototipo que cumple con especificaciones importantes para un sistema tutorial inteligente. El sistema funciona como sistema de consulta y como sistema tutorial.

- Sistema Tutor Inteligente **AUTOTUTOR**.

Autotutor desarrollado por el Tutoring Research Group de la Universidad de Memphis, el cual, apoya a los estudiantes a aprender manteniendo una conversación en un lenguaje natural. Tras décadas de investigación acerca de la tutoría humana se ha creado este Autotutor que implementa estrategias de tutores efectivas promoviendo ganancias significativas en el aprendizaje. Los resultados obtenidos con este Autotutor, apoyan la conclusión de que se trata de un sistema tutorial inteligente efectivo que usa estrategias pedagógicas apropiadas para los aprendices individuales. Presenta una serie de difíciles problemas (o preguntas) que exigen una explicación verbal y el razonamiento en una respuesta.

- Sistema Tutor Inteligente para el Diagnóstico y Tratamiento de Infecciones de Transmisión Sexual (**STIITS**).

STIITS como aplicación constituye el primer Sistema Tutorial Inteligente cubano para el diagnóstico y tratamiento de ITS desarrollado en la provincia de Cienfuegos. STIITS responde a una necesidad social pero también tiene posibilidades de empleo en la docencia. No existe ningún otro software en el país con características similares para el mismo fin.

El objetivo de STIITS como STI es proporcionar una mayor flexibilidad al tutorial manejado por el computador y lograr que este permita una mejor interacción con el estudiante. Con STIITS se pretende capturar el conocimiento de los expertos en las especialidades de ginecología y obstetricia y, crear interacciones en forma dinámica, para una mejor comprensión y desarrollo de estos temas por los estudiantes.

De manera general se puede concluir que en la actualidad los Sistemas Tutores Inteligentes son potencialmente usados en el proceso de autoaprendizaje; proporcionando una formación personalizada, centrándose en las necesidades reales de los estudiantes. Los diferentes sistemas anteriormente mencionados usan estrategias pedagógicas y se caracterizan por la inclusión de experiencia adicional relacionada con el entorno de aprendizaje del estudiante, métodos y técnicas de enseñanza, actuando así como entrenadores personales. Por estas razones es necesario que el Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas incorpore muchas de las características que poseen estas aplicaciones, destacándose la guía que se les brinda a los estudiantes mientras resuelven los ejercicios, la posibilidad de interactuar con el sistema sin que éste se imponga en el proceso de enseñanza-

aprendizaje, el uso de las estrategias pedagógicas apropiadas para los aprendices individuales y finalmente lograr que el sistema permita una mejor interacción con el estudiante.

De esta forma se desarrollará un sistema más flexible adaptado a las necesidades y las preferencias de cada alumno en particular, sin dejar de mencionar que será una aplicación innovadora que servirá de gran ayuda en el aprendizaje del idioma inglés.

1.4 Metodologías de desarrollo de Software

La Metodología se define como un conjunto de métodos orientados a seguir una meta propuesta. Son los procesos que organizados en conjunto proporcionan una secuencias a seguir para la obtención del producto final. (7)

Las metodologías de desarrollo de software son un factor importante en la obtención de productos con calidad en el tiempo requerido. En la actualidad existen diversas metodologías de desarrollo, con características comunes pero también con significativas diferencias, recomendándose un estudio para la correcta selección de la misma, garantizando el éxito de un proyecto de software. (8)

Las ventajas de utilizar una metodología son: (8)

Mejora de los procesos de desarrollo.

- Todas las personas trabajan para un marco común.
- Estandarización de conceptos, actividades y nomenclaturas.
- Actividades de desarrollo apoyadas por procedimientos y guías.
- Resultados del desarrollo predecibles.
- Uso de herramientas de ingeniería software.
- Planificación de actividades en base a un conjunto de tareas definidas y a la experiencia en otros proyectos.
- Recopilación de mejores prácticas para proyectos futuros.

Mejora de los productos.

- Se asegura que los productos cumplen con los objetivos de calidad propuestos.
- Detección temprana de errores.
- Se garantiza la trazabilidad de los productos a lo largo del desarrollo.

Mejora de las relaciones con el cliente.

- El cliente percibe el orden en los procesos.
- Facilita al cliente el seguimiento de la evolución del proyecto.
- Se establecen mecanismos para asegurar que los productos desarrollados cumplen con las expectativas del cliente.

A continuación realizaremos un estudio de las posibles metodologías a utilizar en el desarrollo del sistema a construir.

1.4.1 Programación Extrema

La Programación Extrema (PX), mejor conocida por su nombre en inglés Extreme Programming (XP), es una de las llamadas Metodologías Ágiles de desarrollo de software más exitosa de los tiempos recientes. Esta se basa en la simplicidad, la comunicación y la reutilización del código desarrollado. Es utilizada en proyectos con equipos de desarrollo pequeños y con plazos de entrega corto, de manera que se pueda llevar a cabo una programación rápida o extrema. Una particularidad que tiene, es que el usuario final debe ser miembro del equipo.

Fases de XP (Ver anexo 1)

Entre las principales **características** que distinguen la metodología XP de otras se encuentran: (9)

Es **Iterativa**: El proyecto se divide en versiones y estas a su vez en ciclos. En cada ciclo se desarrollan historias de usuarios que añaden incrementalmente funcionalidad al sistema.

Es **Compleja**: XP propone un cunjunto de técnicas (denominadas prácticas) para ser utilizadas durante el desarrollo. Cada técnica por separado aporta escaso valor al proyecto, algunas son complejas y de difícil implantación.

Es **informal**: XP concede un elevado grado de libertad en la ejecución de los procesos. En lugar de establecer procesos formales de trabajo, se limita a señalar los principios y valores que deben guiar la actuación del equipo. Es característica también la ausencia de documentación.

Cuando usar XP: (10)

- Cuando los clientes no tienen ideas claras de los requisitos y los van cambiando.
- No es apto para proyectos con mucho personal (entre 2 y 10 programadores).
- Para proyectos de riesgo: fecha fija de entrega, algo nunca hecho por el grupo, algo nunca hecho por la comunidad de desarrolladores.

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 del ciclo de la programación.

1.4.2 SCRUM

El Scrum es un modelo de referencia, que define un conjunto de prácticas y roles, que pueden tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. El resultado final en esta metodología se consigue de forma iterativa e incremental. Al comienzo de cada iteración (sprint) se determina que partes se van a construir, tomando como criterios la prioridad para el negocio, y la cantidad de trabajo que se podrá abordar durante la iteración. (11)

Está especialmente indicada para proyectos con un rápido cambio de requisitos. SCRUM entiende el desarrollo de software como algo muy complejo, sujeto a multitud de variaciones e incertidumbres, sobre todo al comienzo. También trata que todos los involucrados en el proyecto conozcan perfectamente en qué punto del desarrollo se encuentra el proyecto y qué falta por hacer. Es ideal para proyectos con un constante cambio de requisitos.

Los **actores** de SCRUM son: (12)

- Product Owner.
- Scrum Master.

- Scrum Team.
- Usuario o Clientes.

Las **acciones** fundamentales de SCRUM son:

- Product Backlog.
- Sprint Backlog.
- Daily Scrum Meeting.

Trabajar bajo la metodología SCRUM ofrece las siguientes ventajas y desventajas: (13)

Ventajas:

- Ofrece la posibilidad de ajustar la funcionalidad en base a la necesidad de negocio del cliente.
- Brinda la visualización del proyecto día a día.

Desventajas:

- No genera toda la evidencia o documentación de otras metodologías.
- Es necesario complementarlo con otros procesos (XP).

1.4.3 Proceso Unificado de Desarrollo

Las siglas RUP en inglés significa Rational Unified Process (Proceso Unificado de Rational) es un producto del proceso de ingeniería de software que proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización del desarrollo. Su meta es asegurar la producción del software de alta calidad que resuelve las necesidades de los usuarios dentro de un presupuesto y tiempo establecido.

Presenta como características esenciales:

En RUP las actividades de desarrollo están **dirigidas por los casos de usos**, esto se refiere a la utilización de los Casos de Usos para el desenvolvimiento y desarrollo de las disciplinas con los artefactos, roles y actividades necesarias.

El Proceso Unificado del Rational es un proceso **iterativo e incremental** que plantea la implementación del proyecto a realizar en iteraciones, con lo cual se pueden definir objetivos por cumplir en cada iteración y así poder ir completando todo el proyecto iteración por iteración. Como parte del enfoque iterativo se encuentra la flexibilidad para acomodarse a nuevos requisitos o a cambios tácticos en los objetivos del negocio.

El desarrollo bajo el Proceso Unificado está **centrado en la arquitectura**. Este proceso se centra en establecer una arquitectura de un sistema y una arquitectura ejecutable construida como un prototipo evolutivo. Este diseño arquitectónico sirve como una solida base sobre la cual se puede planificar y manejar el desarrollo de software basado en componentes. (14)

RUP divide el proceso en cuatro fases:

- **Inicio:** En esta primera fase se define el ámbito y objetivos del proyecto, así como las funcionalidades y capacidades del producto.
- **Elaboración:** En esta fase se define una arquitectura básica y se planifica el proyecto considerando recursos disponibles.
- **Construcción:** Esta fase proporciona un producto construido junto con la documentación basada en la arquitectura de la línea base.
- **Transición:** Se libera el producto y se entrega al usuario para un uso real.

En esta metodología se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo, los cuales son Modelamiento del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba o Testeo, Instalación, Administración de Configuración y Cambios y Ambiente.

Dimensiones de RUP (Ver anexo 2)

1.5 Selección de la metodología a utilizar

Se realizó un estudio de las metodologías más usadas para el desarrollo del software y de acuerdo con las características y complejidad de este sistema, se decide trabajar con RUP. En la presente investigación se realiza el Análisis y Diseño de una propuesta de un Tutor Virtual de Evaluación para el

Aprendizaje Autónomo de Idiomas que será implementada por los programadores. Esto implica que se necesita una gran cantidad de documentación, que permita a los desarrolladores entender con profundidad las funcionalidades que deben implementar. RUP genera una documentación detallada de todo el proceso de software desde sus inicios, a diferencia de metodologías ágiles como XP y SCRUM.

La aplicación a construir tiene objetivos bien definidos y requisitos estables cuya dinámica se mantendrá a lo largo del proceso de desarrollo de la misma; estos se encuentran en proceso de documentación.

RUP resulta más adaptable para los proyectos a largo plazo que necesiten un desarrollo iterativo capaz de mantener un buen control sobre los cambios, además detecta la mayor cantidad de riesgos en las primeras fases y se basa en las mejores prácticas de la Ingeniería de Software.

En comparación con XP hay que señalar que esta última, impone la presencia obligatoria de un representante del cliente todo el tiempo en el desarrollo de software, algo muy improbable en el caso del sistema a desarrollar.

RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización, lo que posibilita su adaptación y configuración al presente proyecto. Es una metodología robusta y bien definida que posibilita la programación orientada a objetos y permite llevar a cabo un proceso de desarrollo práctico, brindando amplias guías, y ejemplos para todas las actividades.

1.6 Lenguajes de modelado

1.6.1 UML (Lenguaje Unificado de Modelado)

Hoy en día para el desarrollo de un software es necesario contar con un plan bien organizado. Un cliente tiene que comprender qué es lo que hará un equipo de desarrolladores; además tiene que ser capaz de señalar cambios si no se han captado claramente sus necesidades. Para ello, todo el proceso de software debe representarse de manera sencilla que permita a los usuarios entender su estructura, contenido y organización. La clave está en organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. UML proporciona tal organización, pues permite a los creadores de sistema generar diseños que capten sus ideas de una forma fácil de entender y de comunicar a otras personas.

UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se le conoce como modelo. Un modelo UML describe lo que supuestamente hará un sistema, pero no dice como implementar dicho sistema.

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar cada una de las partes que comprende el desarrollo de software. Posee formas de modelar conceptos como son los procesos de negocio y funciones de sistema, además de aspectos concretos como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. (15)

1.6.2 BPMN (Business Process Modeling Notation)

BPMN 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 que provee una representación gráfica (mediante diagramas) para expresar 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. (16)

BPMN define un Business Process Diagram (BPD), que se basa en una técnica de grafos de flujo para crear modelos gráficos de operaciones de procesos de negocio. Un modelo de procesos de negocio es una red de objetos gráficos, que son actividades (trabajo) y controles de flujo que definen su orden de rendimiento. Un BPD se estructura con un grupo de elementos gráficos.

Las cuatro **categorías básicas** de elementos son:

- Objetos de flujo.
- Objetos de conexión.
- Artefactos.
- Calles.

Cada una de estas categorías posee determinados elementos gráficos. Los objetos de flujo pueden ser eventos, actividades y compuertas. Los objetos de conexión pueden ser líneas de secuencia,

asociaciones y líneas de mensaje. Los artefactos pueden ser objetos de datos, grupos y anotaciones. Los canales o calles pueden ser simple (pools) o poseer varias pistas (Lanes). (17)

Ventajas

- Considera un único diagrama para la representación de los procesos (BPD, Business Process Diagram).
- Pensado para ser asignado con naturalidad a lenguajes de ejecución.

Desventajas

- Existe un grado importante de ambigüedad en el proceso de negocio modelado.
- No se utilizan todas las potencialidades de modelamiento necesarias para disminuir el grado de ambigüedad.
- El cliente no garantiza el cumplimiento de los estándares mínimos de modelado de proceso. (18)

1.7 Selección del lenguaje de modelado a utilizar

Después de analizar los lenguajes de modelado mencionados anteriormente se llega a la conclusión que el más adecuado para el desarrollo del sistema es UML pues permite modelar el negocio de manera clara y entendible para el equipo de desarrollo. Además permite modelar todo el ciclo de desarrollo de software incluyendo esquemas de bases de datos y componentes reusables, los cuáles ahorrarán tiempo de trabajo a los programadores.

1.8 Herramientas CASE

En las últimas décadas se han creado varias herramientas para el desarrollo de la Ingeniería de Software con el objetivo de desarrollar programas. Las herramientas CASE (Computer Aided Software Engineering) son utilizadas para la automatización del desarrollo de software, contribuyendo así a elevar la productividad y la calidad en el desarrollo de los sistemas informáticos, desde la planificación pasando por el análisis y diseño, hasta la generación del código fuente de los programas y la documentación. (19)

Dentro de las herramientas CASE orientadas a UML podemos encontrar: Visual Paradigm, Rational Rose y ArgoUML.

1.8.1 Visual Paradigm

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado ayuda a una construcción más rápida de aplicaciones de calidad, mejores y a un menor coste, además permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, así como una serie de tutoriales, demostraciones interactivas y proyectos. (20) Es importante señalar que la herramienta de modelado Visual Paradigm no es gratuito, pero la compañía Visual Paradigm UML Comunity, tiene disponible distintas versiones y facilita licencias especiales para fines académicos, sin interés de lucro. (21)

Las razones más importantes por la que una organización debe seleccionar VP-UML sobre otras herramientas de modelado son: (22)

- Permite incorporar dibujos de Visio en cualquier diagrama UML: A diferencia de otras herramientas CASE-UML, VP-UML prolonga esta limitación permitiendo incorporar dibujos de Visio en cualquier diagrama de UML, pudiendo modelar un dominio específico de hardware, software, redes, componentes, etc.

- Completa integración con Microsoft Office: VP-UML soporta la copia de diagramas como objetos OLE, para poder pegar el diagrama a cualquier documento Word, Excel, Power Point. El contenido del diagrama se puede editar directamente, sin embargo no hay que preocuparse de perder el diagrama original de la fuente.

- Herramientas efectivas y costeables: Son las más asequibles con las actuales funciones de las herramientas CASE-UML. Cuestan la octava parte del precio de otras herramientas de modelado con funcionalidades similares. Sus diversas ediciones se encuentran disponibles y se puede elegir de acuerdo a su presupuesto y necesidades.

- Soporta múltiples plataformas: No importa el Sistema Operativo que esté en uso, VP-UML se puede ejecutar en equipo y está disponible en muchas plataformas como Windows, Linux, Mac OS X y Java Desktop.

- Diseño automático del diagrama: Con su diseño automático se pueden ordenar los diagramas desordenados con tan sólo unos pocos clics con el ratón. Se puede elegir el trazado ortogonal, la disposición jerárquica o el árbol del diseño de acuerdo a la naturaleza del diagrama. Cada estilo del diseño puede afinarse con un conjunto de parámetros configurables.

Visual Paradigm es una herramienta de modelado multiplataforma que brinda una interfaz de usuario amigable, permite integrarse con varios IDE de desarrollo. También posibilita la aplicación de Ingeniería inversa, permitiendo entregar al cliente un producto bien documentado. Además tiene incluido un panel para el diseño de la interfaz de usuario lo cual permite no tener que integrarlo con Visio.

1.8.2 Rational Rose

Es una de las más poderosas herramientas CASE de modelado visual para el análisis y diseño de sistemas basados objetos. Se utiliza para modelar un sistema antes de proceder a construirlo. Cubre todo el ciclo de vida de un proyecto.

Rational Rose tiene las **características** siguientes: (23)

- Soporte de ingeniería Forward y/o reversa para algunos de los conceptos más comunes de Java 1.5.
- Capacidad de análisis de calidad de código.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requisitos de aplicación a través de diseños lógicos y físicos.
- Integración con otras herramientas de desarrollo de Rational.
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo.

Desventajas:

- Necesidad de alta capacidad de procesamiento.
- Existen varios lenguajes de programación que no soporta o que sólo lo hace a medias.

1.8.3 ArgoUML

ArgoUML es una herramienta Para modelar sistemas, mediante el cual se realizan diseños en UML "Lenguaje de Modelado Unificado", basada en Java. Puede crear la mayoría de los diagramas estándares de UML.

ArgoUML posee diversas características entre las que se encuentran: (24)

- Las ayudas de ArgoUML abren extensiones UML, XMI, SVG, OCL y otros de los estándares. En este aspecto, ArgoUML todavía está delante de muchas herramientas comerciales.
- ArgoUML es 100% puro de Java. Esto permite que ArgoUML funcione en todas las plataformas para las cuales un puerto confiable de la plataforma Java2 esté disponible.
- ArgoUML es un proyecto abierto. La disponibilidad del código fuente asegura que una nueva generación de diseñadores y de investigadores de software tenga un marco probado del cual puedan conducir el desarrollo y la evolución de las tecnologías de la herramienta CASE.

ArgoUML posee también inconvenientes como son: (25)

- Es muy pesado y requiere de mucha memoria principal.
- Utiliza UML 1.4, la interfaz no es tan agradable ni dispone de íconos para conocer los estereotipos.
- Ni dispone de ingeniería inversa para PHP.

1.9 Selección de la herramienta de modelado a utilizar

Después de analizar las herramientas mencionadas anteriormente se llega a la conclusión que la más adecuada para el desarrollo del sistema es Visual Paradigm pues permite modelar diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Además de las facilidades que brinda para integrarse con el modelo UML, permitiendo un modelado de procesos de negocio capaz de plasmar en modelos, las actividades reales del Tutor Virtual.

La herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto, lo cual es la intención del sistema a desarrollar. Incorpora el soporte para trabajo en equipo, que permite

que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros.

Esta herramienta es multiplataforma, es decir, que puede ser utilizada en cualquier sistema operativo; mientras que el Rational sólo se puede utilizar en Windows. Además Visual Paradigm puede soportar el ciclo de vida completo de un software, permitiendo construir aplicaciones de calidad a un menor costo.

1.10 Lenguajes de Programación

Para el desarrollo del STI se analizaron diferentes alternativas de desarrollo. Desde la utilización de un único lenguaje como PHP utilizando sus grandes ventajas, hasta lenguajes que no dependieran en absoluto de la máquina de ejecución como pude ser Java. Pero sin perder de vista que el objetivo fundamental, es la obtención de un lenguaje que fuese multiplataforma, libre y de fácil uso para los programadores.

Los lenguajes de programación más utilizados en la actualidad son Java, PHP y C++.

1.10.1 Java

Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ es un lenguaje que adolece de falta de seguridad, pero C y C++ son lenguajes más difundidos, por ello Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje. Este elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles. En la actualidad su uso es promovido para el desarrollo de aplicaciones empresariales del lado del servidor, especialmente a través del estándar J2EE, así como en dispositivos móviles(a través del estándar J2ME). (26)

Ventajas:

- Totalmente orientado a objetos.
- Es independiente de la plataforma.

Desventajas:

- Necesita una máquina virtual.

- Utiliza grandes cantidades de memoria.

1.10.2 C++

Es un lenguaje imperativo orientado a objetos derivado del C. En realidad un superconjunto de C, que nació para añadirle cualidades y características de las que carecía. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción.

La definición "oficial" del lenguaje nos dice que C++ es un lenguaje de propósito general basado en el C, al que se han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones online, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de librería (en realidad la librería Estándar C es un subconjunto de la librería C++). Por lo general puede compilarse un programa C bajo C++, pero no a la inversa si el programa utiliza alguna de las características especiales de C++. (27)

Ventajas:

- Es muy veloz.
- Se permite escribir código en C.

Desventajas:

- C++ es compilado y se produce un ejecutable válido sólo para una plataforma concreta.
- Es inseguro.
- No es apropiado para la web.

1.10.3 PHP

El PHP (acrónimo de PHP: Hypertext Preprocessor), es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. El PHP inició como una modificación a Perl escrita por Rasmus Lerdorf a finales de 1994.

“El PHP es un lenguaje de script incrustado dentro del HTML. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas”.

El código de PHP es abierto por lo que cuenta de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparen rápidamente. El código se pone al día continuamente con mejoras y extensiones del lenguaje para ampliar las capacidades de PHP. Es utilizado en aplicaciones Web-relacionadas por algunas de las organizaciones más sobresalientes tales como Mitsubishi, Redhat, Der Spiegel, MP3-Lycos, Ericsson y NASA. (28)

Desventajas

- No posee una abstracción de base de datos estándar, sino bibliotecas especializadas para cada motor (a veces más de una para el mismo motor).
- Por su diseño dinámico no puede ser compilado y es muy difícil de optimizar.

Las **ventajas** de PHP son: (29)

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones (desde php5).

1.11 Selección del Lenguaje de Programación a utilizar

Una vez analizadas algunas de las características fundamentales de los lenguajes JAVA, C++ y PHP, se puede concluir que debido a la particularidad y funcionalidades que cada uno de ellos brinda, se ha seleccionado PHP, ya que este lenguaje por sus características específicas es el más óptimo para aplicar a este tipo de trabajo. Es un lenguaje sencillo pero muy potente, multiplataforma, rápido y probado por sus seguidores. PHP puede interactuar con muchos motores de bases de datos como My SQL, MS SQL, Oracle, PostgreSQL, y muchos otros. Este lenguaje es libre, por lo que cuenta con un gran grupo de programadores, lo que hace su uso bastante cómodo, a la hora de detectar errores y fallos de funcionamiento, lo más rápido posible.

1.12 Framework para PHP

1.12.1 Symfony 1.2.8

Como parte de la selección del equipo de arquitectura, se propuso Symfony 1.2.8, ya que es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL y PostgreSQL. Se puede ejecutar tanto en plataformas Linux, como en plataformas Windows. (30)

Desventajas:

- Para su utilización se requiere de la generación de grandes cantidades de código y una configuración exhaustiva de antemano.
- Precisa de una estructuración estricta de sus directorios que se establece por consola a la hora de crear módulos.

Ventajas:

- Funciona correctamente en cualquier tipo de plataforma.
- Independiente del sistema gestor de bases de datos. Su capa de abstracción y el uso de Propel, permiten cambiar con facilidad de SGBD en cualquier fase del proyecto.
- Aunque utiliza MVC (Modelo vista controlador), tiene su propia forma de trabajo en este punto, con variantes del MVC clásico como la capa de abstracción de base de datos, el controlador frontal y las acciones.

1.13 Gestores de Base de Datos

Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a realizar las siguientes acciones:

- Definición de los datos.
- Mantenimiento de la integridad de los datos dentro de la base de datos.
- Control de la seguridad y privacidad de los datos.
- Manipulación de los datos.

1.13.1 MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca. MySQL es software libre, pero MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL. Este es uno de los gestores de bases de datos más usados en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Las principales **características** de este gestor de bases de datos son las siguientes: (31)

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc.).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

1.13.2 PostGreSQL

PostGreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. PostGreSQL es una derivación libre (OpenSource) de este proyecto, y utiliza el lenguaje SQL92/SQL99. Incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostGreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos. Soporta la gran mayoría de las transacciones SQL, control concurrente, teniendo a su disposición varios "language bindings" como por ejemplo C, C++, Java, Python, PHP y muchos más.

Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. También permite la creación de tipos propios. Incorpora una estructura de datos array y funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes. Permite la declaración de funciones propias, así como la definición de disparadores. Soporta el uso de índices, reglas y vistas. Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales. Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

PostGreSQL carece de un conjunto de herramientas que permitan una fácil gestión de los usuarios y de las bases de datos que contenga el sistema. Por otro lado, la velocidad de respuesta que ofrece este gestor con bases de datos relativamente pequeñas puede parecer un poco deficiente, aunque esta misma

velocidad la mantiene al gestionar bases de datos realmente grandes. Sin embargo PostGreSQL es un gestor magnífico, que posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios web que posean alrededor de 500.000 peticiones por día. (32)

1.14 Selección del Gestor de Base de Datos a utilizar

En la presente investigación se ha seleccionado PostGreSQL como gestor de base de datos a utilizar, ya que constituye una herramienta muy potente que permite grandes volúmenes de información, es software libre, multiplataforma y tiene todas las características de los SGBD modernos.

Además tiene una alta integridad, robustez, y extensibilidad. También brinda gran seguridad y posee una alta escalabilidad, haciéndolo idóneo para sistemas que posean grandes peticiones por día, como es el caso del sistema que se quiere desarrollar.

Conclusiones

En este capítulo se detallaron los principales conceptos relacionados con los STI y los diferentes campos de su aplicación en la actualidad. Se analizaron posibles soluciones existentes en Cuba y en el mundo que implementan funcionalidades propuestas para la aplicación. También se desarrollo el análisis de posibles metodologías de desarrollo a utilizar, así como lenguajes de programación, gestores de bases de datos, herramientas Case y la selección de un lenguaje de modelado. Dejando sentadas las bases para el desarrollo de la solución propuesta seleccionando:

- RUP como metodología de desarrollo, siendo esta una metodología robusta que resulta más adaptable para los proyectos a largo plazo que necesiten un desarrollo iterativo capaz de mantener un buen control sobre los cambios y desarrollar una documentación detallada de todo el proceso de desarrollo de software.
- Visual Paradigm como herramienta de modelado ya que tiene facilidad para integrarse con el modelo UML.
- PostgreSQL como gestor de bases de datos debido a la necesidad de desarrollar un producto multiplataforma lo cual incidirá en la calidad del trabajo del sistema.
- PHP como lenguaje de programación y el framework de desarrollo Symfony 1.2.8, el mismo fue seleccionado luego de un estudio realizado por el equipo de arquitectura.

CAPÍTULO 2: Características del Sistema

Introducción

En este capítulo se realizarán las tareas correspondientes a los procesos de análisis del Tutor Virtual de Evaluación para Aprendizaje Autónomo de Idiomas dándole solución al problema planteado en la investigación, por lo que es necesaria la identificación de los conceptos asociados al Modelo de Dominio, los requisitos funcionales y no funcionales del sistema, así como lo representación de diagramas y descripciones de gran importancia en el desarrollo del proyecto.

2.1 Características del Sistema

Entre las características principales del Tutor Virtual de Evaluación para Aprendizaje Autónomo de Idiomas se encuentran:

- Permitir darle la bienvenida al asistente mediante un módulo Informativo.
- Permitir realizar diferentes diagnósticos y encuestas al usuario mediante un módulo de Diagnóstico General.
- Permitir conocer el nivel de idioma del usuario mediante un módulo de Diagnóstico de Idioma.
- Permitir obtener un perfil del usuario con los datos obtenidos en los dos módulos anteriores.
- Permitir negociar actividades y realizar examen de acreditación a los diferentes usuarios.

Este sistema debe llevar a cabo el procesamiento de la información a partir de un conjunto de encuestas iniciales respondidas por el usuario, además de determinar el estado actual del mismo así como su estilo o estilos de aprendizaje a la vez que genere la ruta de autoaprendizaje a seguir durante el entrenamiento mejorando la calidad en el proceso de formación académica en esta disciplina.

Es importante garantizar un sistema automatizado en línea 24 horas donde el usuario pueda completar una ruta de aprendizaje sin la imprescindible intervención de un profesor (tutor, guía o asesor físico).

2.2 Modelo de Dominio

El Modelo de Dominio (o Modelo Conceptual) es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Define un modelo de clases común para todos los implicados en el desarrollo, representadas en objetos del dominio, sirve como interlocutor entre

clientes y desarrolladores. El propósito fundamental de este modelo es generar una terminología común y sentar las bases del entendimiento del desarrollo y no para definir el sistema completo. Cualquiera sea la solución de casos de uso que se haya elegido, los conceptos e ideas propias del dominio del problema son las mismas; un mismo modelo de dominio contempla cualquiera de las soluciones analizadas. El modelo de dominio es global, es decir se realiza para todos los casos de uso y no para uno en particular.

2.3 Realización del Modelo de Dominio

El entorno es la Universidad de las Ciencias Informáticas en la cual se imparte la docencia de Idiomas Extranjeros mediante el Eva Moodle con modalidad semipresencial, es decir utilizando a los docentes en las aulas, empleando para ello materiales como libros, y los medios de enseñanza tradicionales. La solución que se plantea con la presente investigación está dirigida a los aprendices que desarrollan un proceso de aprendizaje autónomo de idioma, con el fin de favorecer la gestión de la evaluación en este contexto. Al analizar dicho entorno nos percatamos que es imposible determinar procesos de negocio, debido a que no están bien definidos. Actualmente en la universidad no existe un proceso de este tipo que contenga actividades entrelazadas dentro de él mismo, las fronteras de estos no se establecen con claridad y se fusionan constantemente. Se determinan algunos eventos y conceptos que participan de dicho entorno como son:

- Orientación de contenidos.
- Explicación de contenidos.
- Preparación independiente de los contenidos.
- Comprobación del contenido aprendido y certificación de nivel.

Partiendo de la descripción de los principales conceptos obtenidos (**Ver artefacto Modelo del dominio**), el diagrama del modelo de dominio se estructuró de la siguiente forma:

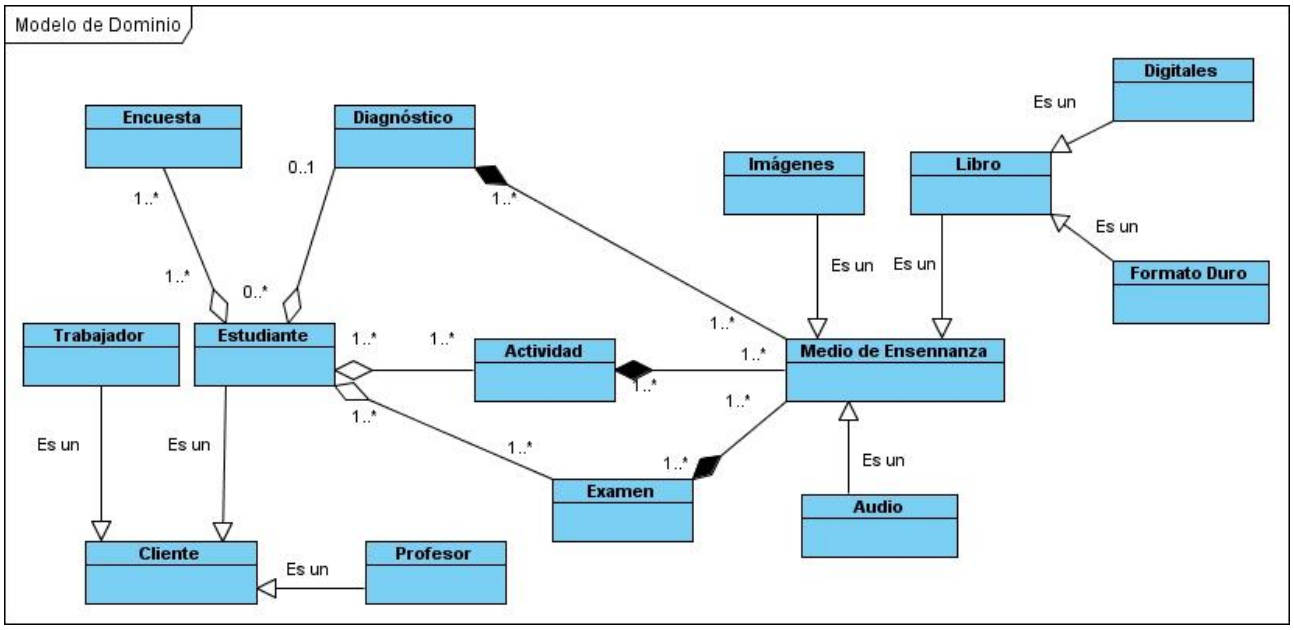


Figura 2: Modelo del dominio.

2.4 Especificación de los Requisitos del Software

El propósito general de la captura de requisitos es obtener una descripción correcta de lo que debe hacer el sistema y delimitar su alcance. Los requisitos juegan un papel importante durante el ciclo de vida de un programa. Durante la fase de inicio se identifican la mayoría de estos y se enumeran a través de requisitos funcionales y no funcionales, las acciones que el sistema deberá ser capaz de realizar.

2.4.1 Requisitos Funcionales

Los requisitos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Son capacidades o condiciones que el sistema debe cumplir.

El Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas cuenta con los siguientes requisitos funcionales, los cuales se encuentran agrupados por funcionalidades, quedando primeramente el caso de uso y los requisitos contenidos dentro de este como se muestra a continuación:

CU 1: Gestionar encuesta.

R 1.1: Diseñar encuesta.

R 1.2: Modificar encuesta.

R 1.3: Eliminar encuesta.

CU 2: Realizar encuestas.

R 2.1: Procesar resultados de la encuesta de estilo.

R 2.2: Procesar resultados de la encuesta de necesidades.

R 2.3: Procesar resultados de la encuesta de fase.

R 2.4: Procesar resultados de la encuesta de nivel.

R 2.5: Procesar resultados de la encuesta de satisfacción.

R 2.6: Generar diagnóstico de idioma.

CU 3: Gestionar pregunta.

R 3.1: Crear pregunta.

R 3.2: Modificar pregunta.

R 3.3: Eliminar pregunta.

R 4: Visualizar diagnóstico.

CU 5: Realizar diagnóstico de idioma.

R 5.1: Procesar diagnóstico de audición.

R 5.2: Procesar diagnóstico de lectura.

R 5.3: Procesar diagnóstico de escritura.

R 5.4: Procesar diagnóstico de expresión oral.

R 5.5: Mostrar nivel de idioma alcanzado.

CU 6: Gestionar actividad con la clasificación.

R 6.1: Configurar actividad.

R 6.2: Modificar actividad.

R 6.3: Eliminar actividad.

CU 7: Realizar contrato.

R 7.1 Mostrar contrato.

R 7.2: Mostrar aviso de fin de contrato.

CU 8: Planificar agenda.

R 8.1: Mostrar días disponibles para acreditación.

R 8.2 Seleccionar fecha de acreditación.

R 8.3 Seleccionar hora de acreditación.

CU 9: Gestionar agenda.

R 9.1: Crear agenda.

R 9.2: Modificar agenda.

R 9.3: Eliminar agenda.

R 9.4: Visualizar agenda.

CU 10: Realizar examen de acreditación.

R 10.1: Seleccionar actividades de examen de acreditación.

R 10.2: Generar examen de acreditación.

R 10.3: Procesar respuestas del examen de acreditación.

R 10.4: Incorporar resultados de examen de acreditación al perfil.

CU 11: Autenticar usuario.

R 11.1: Autenticar.

R 11.2: Asignar rol.

R 11.3: Mostrar usuarios en línea.

CU 12: Gestionar rol.

R 12.1: Diseñar rol.

R 12.2: Modificar rol.

R 12.3: Eliminar rol.

CU 13: Gestionar regla.

R 13.1: Crear regla.

R 13.2: Modificar regla.

R 13.3: Eliminar regla.

R 13.4: Visualizar regla.

CU 14: Gestionar inciso.

R 14.1: Crear inciso.

R 14.2: Modificar inciso.

R 14.3: Eliminar inciso.

R 14.4: Visualizar inciso.

CU 15: Gestionar Diagnóstico.

R 15.1: Configurar diagnóstico.

R 15.2: Modificar diagnóstico.

R 15.3: Eliminar diagnóstico.

CU 16: Gestionar Evaluación.

R 16.1: Asignar evaluación.

R 16.2: Modificar evaluación.

R 16.3: Eliminar evaluación.

2.4.2 Requisitos No Funcionales

Requisitos de Software:

- El servidor central de procesamiento deberá tener instalado:
  - ✓ Sistema Operativo Windows o Linux.

- ✓ Framework Symfony 1.2.8.
- ✓ Servidor WAMP 2.0 (Windows) o LAMP (Linux).

- ➢ Las PC Clientes deberán tener instalado:
  - ✓ Sistema Operativo Windows o Linux.
  - ✓ Navegadores Web Internet Explorer 5.0 o Mozilla Firefox 3.5.
- ➢ El servidor central de bases de datos deberá tener instalado:
  - ✓ Sistema Operativo Windows o Linux.
  - ✓ Gestor de Base de Datos PostgreSQL 8.3.
- ➢ El servidor de respaldo de bases de datos deberá tener instalado:
  - ✓ Sistema Operativo Windows o Linux.
  - ✓ Gestor de Base de Datos PostgreSQL 8.3.

Requisitos de Hardware:

- ➢ Todas las computadoras deberán tener una tarjeta de red con una velocidad de transmisión de 100Mbps.
- ➢ El Servidor central de procesamiento requiere:
  - ✓ Memoria RAM 1 GB DDR2 (Double Data Rate) (óptimo 8 GB).
  - ✓ Procesador de 1.90GHZ.
  - ✓ 80 GB de capacidad de almacenamiento mínimo (óptimo 250 GB).
- ➢ Las PC Clientes requieren:
  - ✓ Memoria RAM 128MB DDR2 (óptimo 2 GB).
  - ✓ Procesador de 1.90GHZ.

- El servidor central de bases de datos requiere:
  - ✓ Memoria RAM 1 GB DDR2 (óptimo 8 GB).
  - ✓ Procesador de 1.90GHZ.
  - ✓ 80GB de capacidad de almacenamiento mínimo (óptimo 1 TB).
- El servidor de respaldo de bases de datos requiere:
  - ✓ Memoria RAM 1 GB DDR2 (óptimo 8 GB).
  - ✓ Procesador de 1.90GHZ.
  - ✓ 80 GB de capacidad de almacenamiento mínimo (óptimo 1 TB).

Consultar los restantes requisitos No Funcionales en el artefacto Especificación de requisitos de software.

A continuación se muestra una tabla donde se elabora un resumen de los requisitos funcionales y no funcionales, clasificando los requisitos funcionales de acuerdo a su prioridad, obteniéndose 54 requisitos clasificados de alta prioridad, debido a la necesidad que tenía el usuario de implementarlos rápidamente.

Por otra parte se clasificaron los requisitos no funcionales de la siguiente manera: orientados al usuario (Seguridad, Usabilidad, Disponibilidad), orientados al cliente (Software, Hardware) y orientados a los desarrolladores (Disponibilidad, Portabilidad). Se obtuvo finalmente un total de 65 requisitos.

Tipo de Requisitos	Clasificación	Número
Funcionales	Alta prioridad	54
	Baja prioridad	0
No Funcionales	Orientados al usuario	4
	Orientados al cliente	2
	Orientados a los desarrolladores	5

Total de requisitos	**65**

Tabla 2 Resumen de Requisitos.

2.5 Modelo de Casos de Usos del Sistema

Un modelo de casos de uso es un modelo del sistema que contiene actores, casos de uso y sus relaciones. El modelo de casos de uso permite que los desarrolladores y los clientes lleguen a un acuerdo sobre los requisitos, es decir sobre las condiciones y posibilidades que debe cumplir el sistema. El modelo de casos de uso describe lo que hace el sistema para cada tipo de usuario. Cada usuario se representa mediante uno o más actores.

2.5.1 Actores del Sistema

Los actores representan terceros fuera del sistema que colaboran con el sistema. Al identificar los actores del sistema se identifica el entorno externo del sistema.

Actor del Sistema	Descripción
Usuario	Es aquel que se beneficia con la interacción con la plataforma, ya que le permite realizar el autoaprendizaje, puede ser un estudiante o un trabajador.
Administrador	Es aquella persona que contribuye a la facilitación del autoaprendizaje de los usuarios.
Evaluador	Es aquella persona que contribuye a la facilitación del autoaprendizaje de los usuarios, asignándoles evaluaciones a los usuarios.

Tabla 2 Definición de los actores del sistema.

2.5.2 Casos de Uso del Sistema

Los casos de uso son el componente clave del modelado. Su propósito es ilustrar como un sistema permite a un actor cumplir una meta, ilustrando todos los posibles caminos apropiados que ellos pueden tomar para cumplirla, así como las situaciones que podrían hacerlo fallar.

2.5.3 Diagrama de Casos de Uso del Sistema

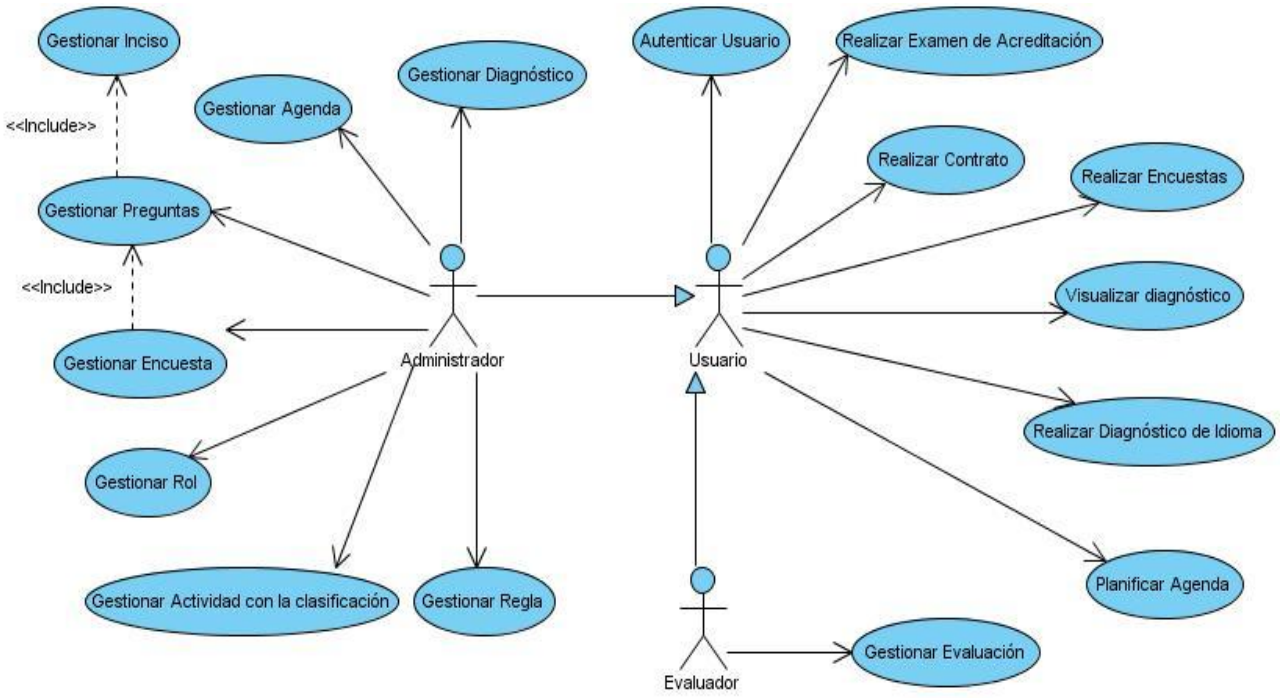


Figura 3 Diagrama de Casos de Uso del Sistema.

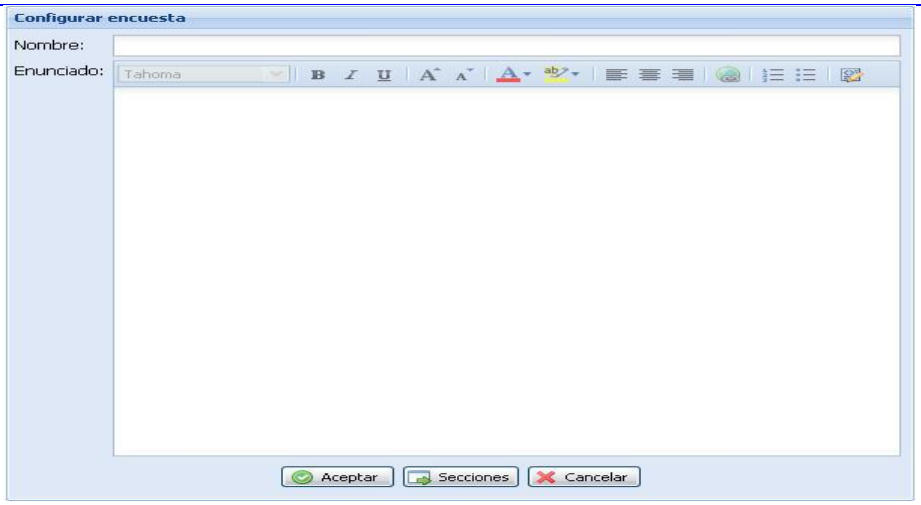
2.6 Especificación textual de los casos de usos

La especificación de los casos de uso contiene las propiedades textuales de los casos de uso, además contiene una descripción del flujo de eventos, describiendo la interacción entre actores y el sistema. La descripción que se muestra a continuación corresponde al caso de uso Gestionar encuesta.

Caso de Uso Gestionar Encuesta.

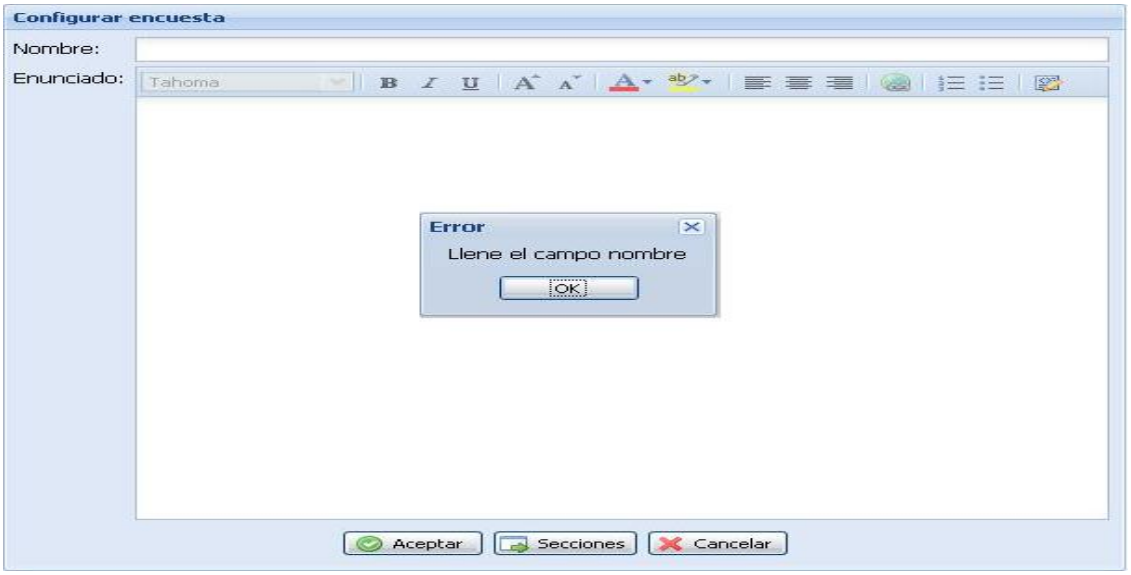
Caso de Uso:	Gestionar Encuesta.
Actores:	Administrador.
Resumen:	El CUS se inicia cuando el actor selecciona la opción Gestionar encuesta, el sistema realiza la acción seleccionada por el actor y muestra las posibles opciones a seleccionar y termina el CUS.
Precondiciones:	1. El actor debe estar autenticado en la aplicación.
Referencias	R 1.1, R 1.2, R 1.3.
Prioridad	Alta

Flujo Normal de Eventos	
Acción del Actor.	**Respuesta del Sistema.**
1. El caso de uso se inicia cuando el actor selecciona la opción Gestionar encuesta.	2. El sistema muestra las opciones: Diseñar encuesta, Modificar encuesta, Eliminar encuesta.
3.a) El actor selecciona la opción Diseñar encuesta, ir a la sección Diseñar encuesta. b) El actor selecciona la opción Modificar encuesta, ir a la sección Modifica encuesta. c) El actor selecciona la opción Eliminar encuesta, ir a la sección Eliminar encuesta.	
Sección "Diseñar Encuesta."	
Acción del Actor.	**Respuesta del Sistema.**
1. El actor selecciona la opción diseñar encuesta.	2. El sistema muestra la interfaz que permitirá configurar la encuesta.
3. El actor introduce los datos necesarios de la encuesta (Nombre: letras y números, Enunciado: letras y números) y presiona la opción aceptar.	4. El sistema verifica los datos introducidos por el actor.
	5. Si los datos están correctos el sistema crea la encuesta.
	6. El sistema va al caso de uso Gestionar pregunta.
	7. Finaliza el caso de uso.
Prototipo de Interfaz.	



Flujo Alterno de eventos.	
Acción del Actor.	**Respuesta del Sistema.**
	1. El sistema muestra un mensaje de error indicando donde está el dato erróneo y retorna a la acción 3.

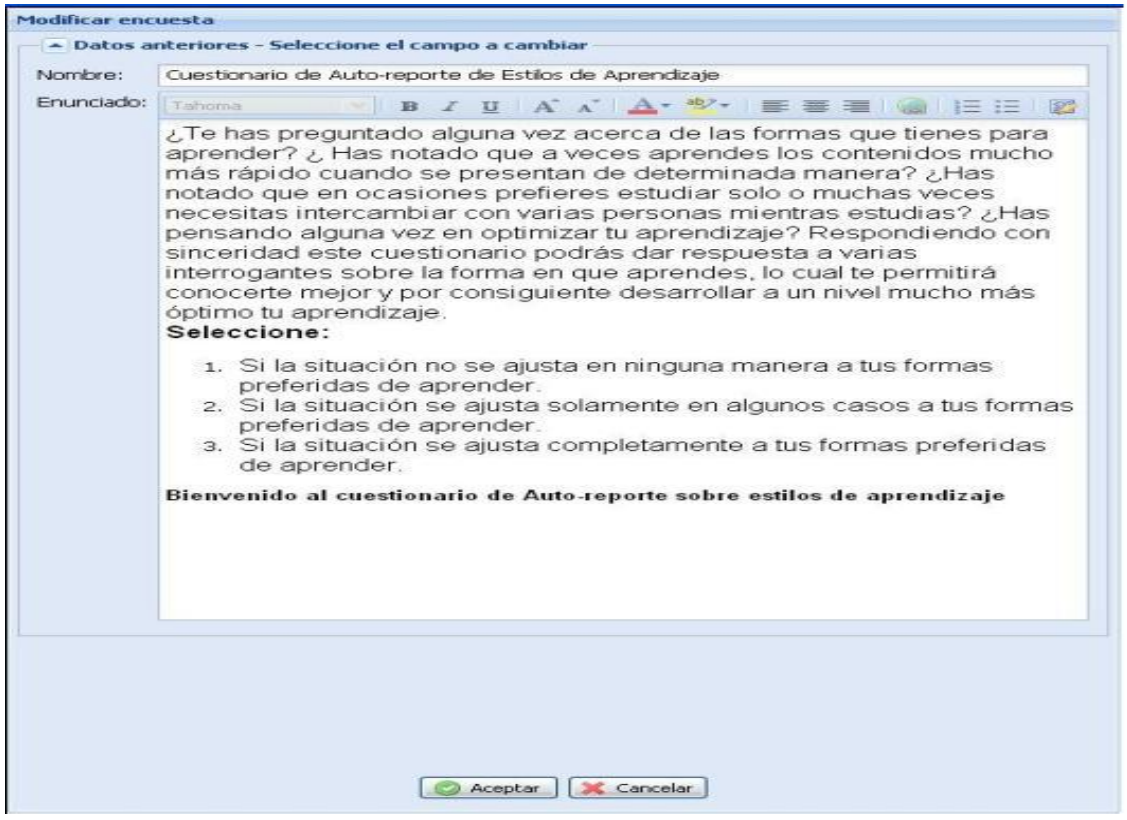
Prototipo de Interfaz



Sección “Modificar Encuesta.”	
Acción del Actor.	**Respuesta del Sistema.**

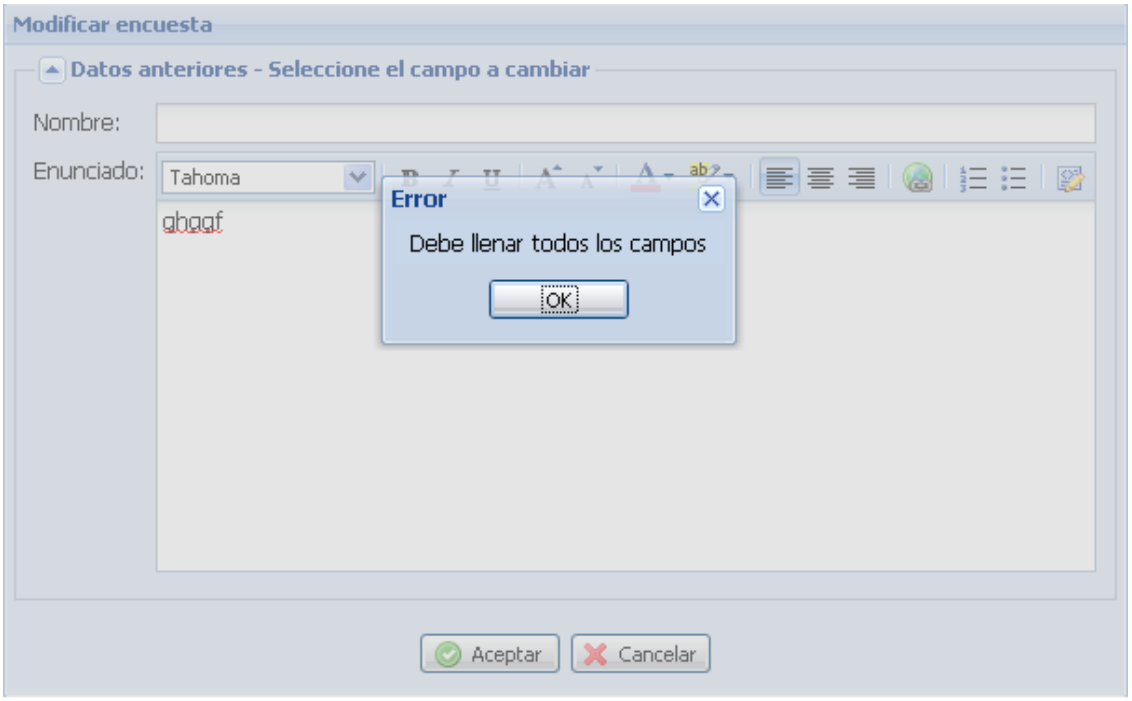
1. El actor selecciona la opción de Modificar encuesta.	2. El sistema muestra todas las encuestas existentes.
3. El actor selecciona la encuesta y presiona la opción aceptar.	4. El sistema muestra la encuesta seleccionada.
5. El actor realiza los cambios necesarios a la encuesta y presiona la opción aceptar.	6. El sistema verifica los datos modificados por el actor.
	7. Si los datos están correctos el sistema modifica los datos de la encuesta seleccionada y termina el CUS, si los datos no están correctos ir al flujo alterno de eventos.

Prototipo de Interfaz.



Flujo Alterno de eventos	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra un mensaje de error indicando donde está el dato erróneo y retorna a la acción 5.

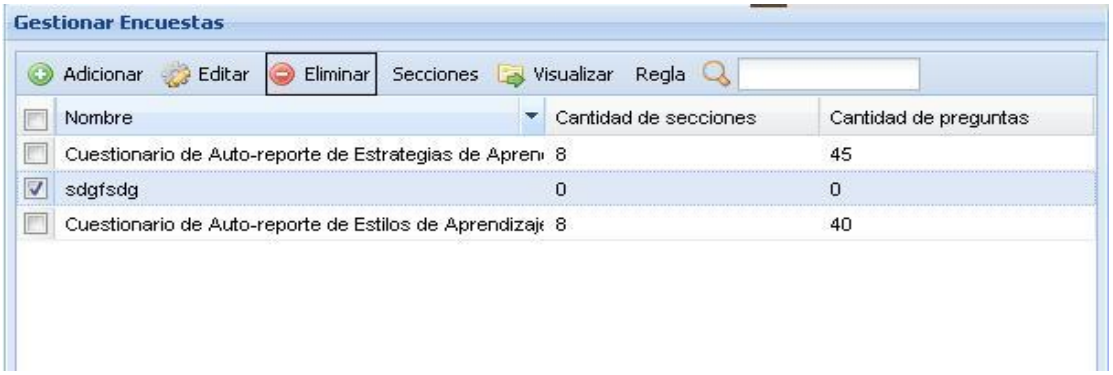
Prototipo de Interfaz.



Sección "Eliminar Encuesta."	
Acción del Actor.	**Respuesta del Sistema.**
1. El actor selecciona la opción eliminar encuesta.	2. El sistema muestra un listado con todas las encuestas existentes.
3. El actor selecciona la encuesta que será eliminada y presiona la opción aceptar.	4. El sistema muestra un mensaje de advertencia para la acción a realizar.
5. El actor confirma si desea o no	6. Si el actor confirma la eliminación, el

eliminar la encuesta.	sistema elimina la encuesta y finaliza el caso de uso, sino confirma la eliminación ir al flujo alterno de eventos.

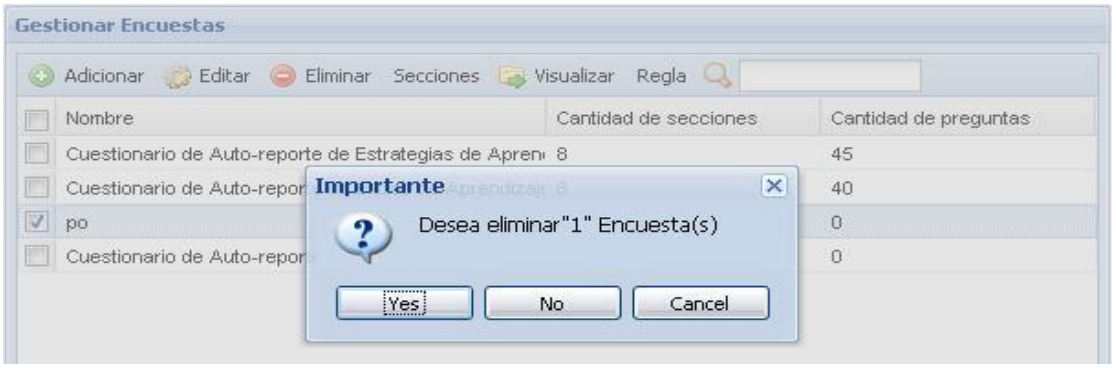
Prototipo de Interfaz



Flujo Alterno de eventos

Acción del Actor	Respuesta del Sistema
	1. Se culmina el CUS sin ejecutar ninguna acción.

Prototipo de Interfaz



Poscondiciones	1. Encuesta creada. 2. Encuesta modificada. 3. Encuesta eliminada.

Consultar las descripciones de los restantes Casos de Uso en el artefacto Modelo del sistema.

Conclusiones

En este capítulo se presentaron las principales características del sistema. Se capturaron los objetos más importantes para comprender el funcionamiento del sistema a través del Modelo de Dominio, definiéndose cada uno de los conceptos relacionados en el mismo. Además se realizó el levantamiento de requisitos para identificar los requisitos funcionales y no funcionales que debe presentar la aplicación. Se realizó el diagrama de Casos de Uso del Sistema y fueron realizadas las descripciones textuales correspondientes a cada caso de uso lo cual provee de una visión general de qué es lo que el sistema debe hacer, para de esta forma darle solución a los problemas planteados.

CAPÍTULO 3: Análisis y Diseño del Sistema

Introducción

En este capítulo se realizan actividades referentes al análisis y diseño del sistema propuesto, el análisis y el diseño es un flujo de trabajo que tiene como propósito transformar los requisitos tanto funcionales como no funcionales en un diseño de clases concibiendo las relaciones e interacciones que presentan entre ellas. Se construyen los diagramas de clases del análisis y del diseño, se presenta el diagrama de colaboración correspondiente a cada caso de uso del sistema y además se define la arquitectura y los patrones de diseño a utilizar.

3.1 Análisis

Las actividades del análisis son desarrolladas con el objetivo de facilitar la entrada al diseño, por lo que son un paso inicial y una primera aproximación conceptual para aumentar el nivel de especificidad en aras de garantizar el cubrimiento de los requisitos funcionales y obtener una visión de qué hace el sistema. El modelo de análisis está compuesto por artefactos como: clases del análisis y realizaciones de casos de uso del análisis.

Las clases del análisis van a representar abstracciones de conceptos, en las cuales deben incluirse atributos y operaciones a un nivel alto. La realización de casos de uso del análisis describe cómo se lleva a cabo y se ejecuta un caso de uso determinado en términos de las clases del análisis y de sus objetos en interacción.

3.1.1. Diagrama de Clases del Análisis.

El diagrama de clases del análisis (DCA) está compuesto por clases del análisis y sus relaciones. Estas clases están centradas en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos importantes y relaciones del dominio. Las mismas se clasifican en: Clases de interfaz, Clases de Control, Clases de Entidad.

Seguidamente se muestra el diagrama de clases del análisis correspondiente al caso de uso Gestionar encuesta, **consultar los restantes diagramas en el artefacto Modelo del análisis**.

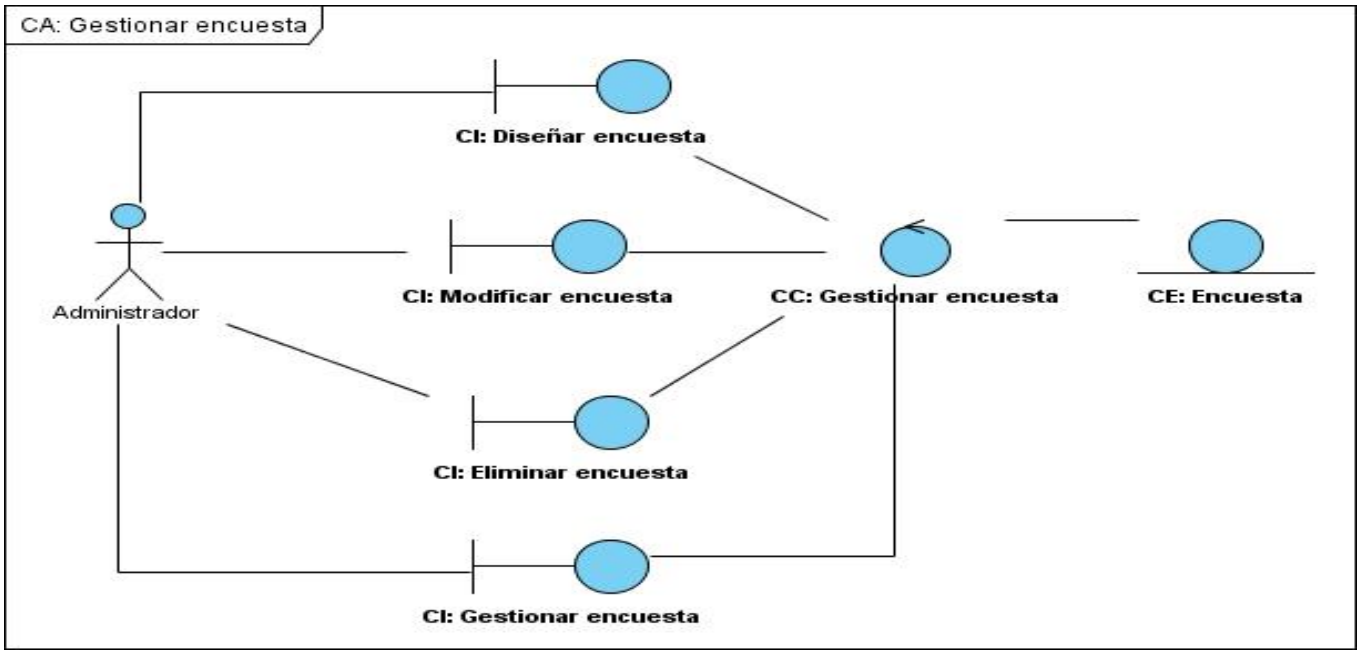


Figura 4: Diagrama de Clases del Análisis. CU Gestionar_encuesta.

3.1.2. Realización de casos de uso del análisis.

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un módulo de un sistema, mostrando el conjunto de objetos que participan y sus relaciones, incluyendo los mensajes que intercambian entre ellos. Los diagramas de interacción se clasifican en dos tipos: los diagramas de Colaboración y los diagramas de Secuencia.

De dichos diagramas se escogieron los de colaboración, los cuales muestran cómo las instancias específicas de las clases trabajan juntas para alcanzar un objetivo común. A continuación se presenta el diagrama de colaboración correspondiente al caso de uso crítico Gestionar encuesta.

Consultar los diagramas de los restantes casos de uso en el artefacto Modelo de diagramas de colaboración.

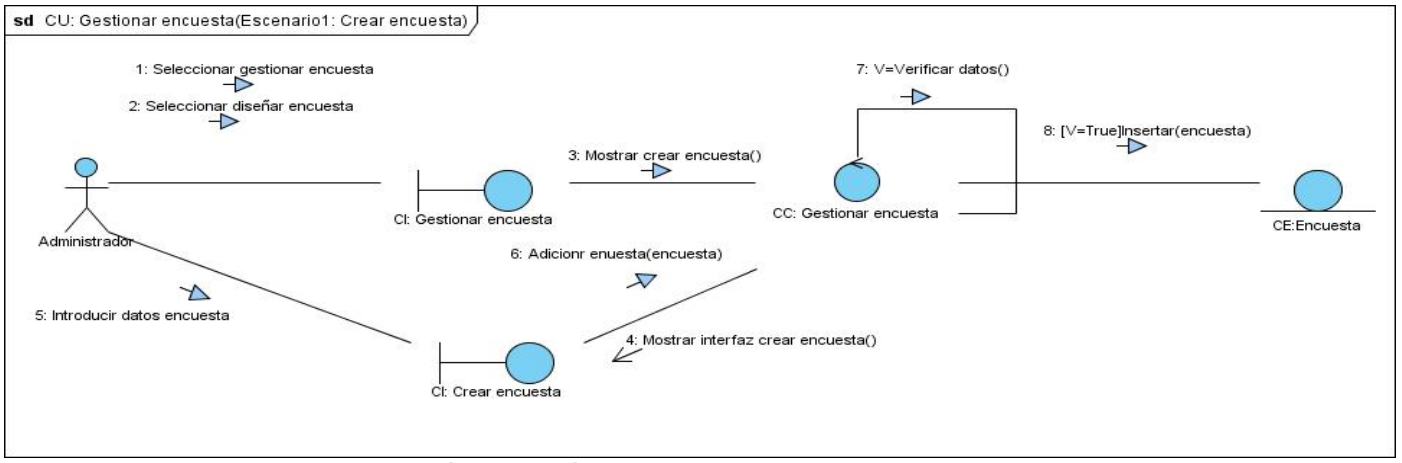


Figura 5: Diagrama de Colaboración del Análisis. CU Gestionar encuesta (Escenario Crear).

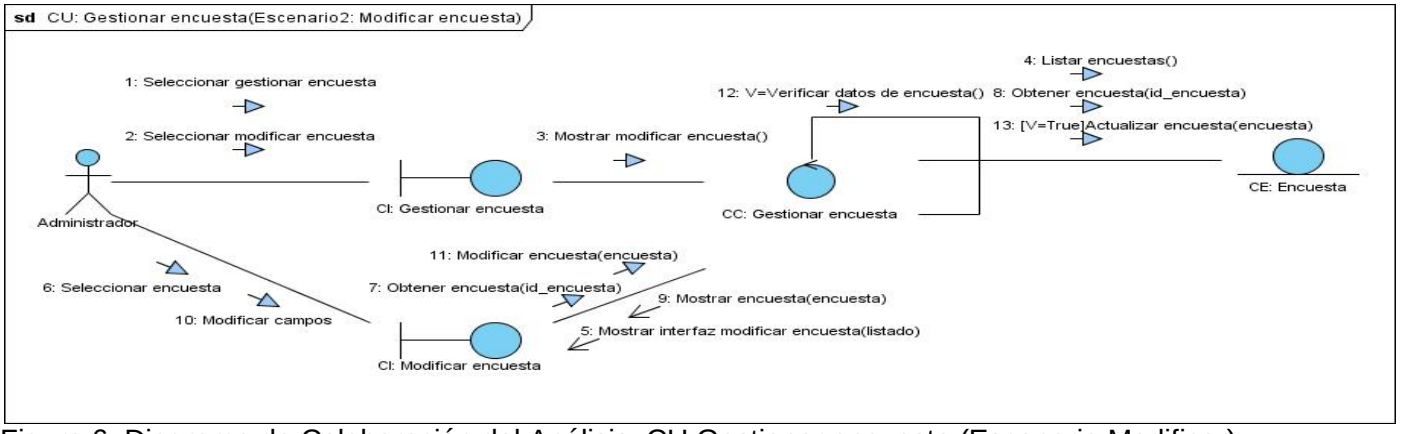


Figura 6: Diagrama de Colaboración del Análisis. CU Gestionar encuesta (Escenario Modificar).

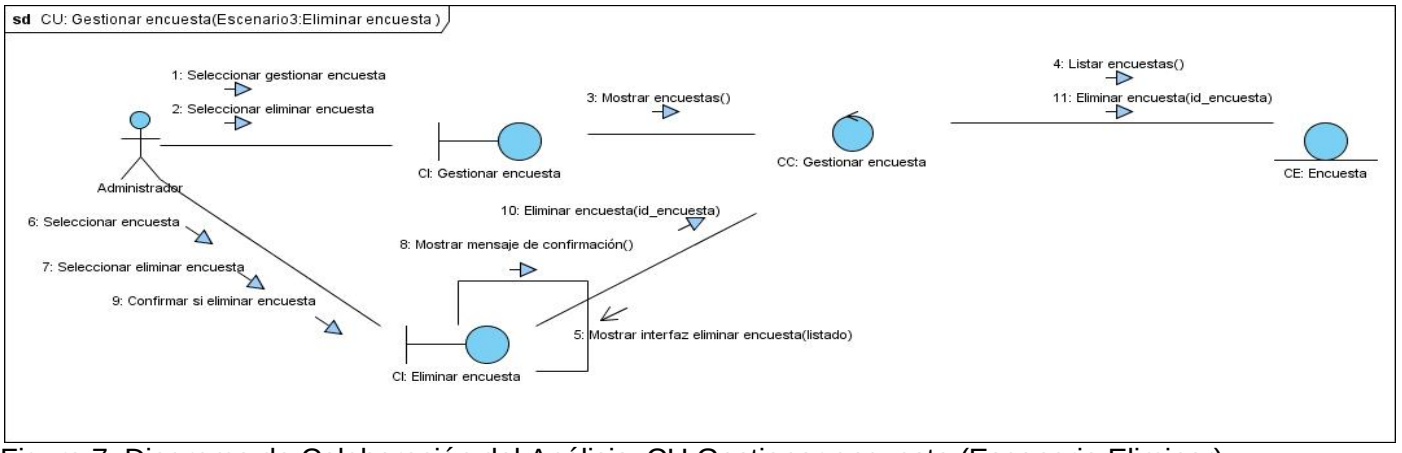


Figura 7: Diagrama de Colaboración del Análisis. CU Gestionar encuesta (Escenario Eliminar).

3.2 Diseño

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, o sea cómo cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. En el diseño se modela el sistema y se propone una arquitectura que soporte todos los requisitos.

3.2.1 Arquitectura definida para el sistema.

La arquitectura de software establece los fundamentos para que el equipo de desarrollo trabaje en una línea común y alcanzar así los objetivos del sistema, es la estructura de un sistema, la base de una aplicación. En ésta se analiza la aplicación desde varios puntos de vista y tiene la responsabilidad de definir los módulos principales y las relaciones entre estos. En la arquitectura aparecen los artefactos más importantes y diferentes, para establecer un esquema de cómo deben ser los próximos artefactos a construir. De obtenerse un artefacto demasiado diferente a los demás, éste formaría parte de la arquitectura. En RUP, ésta se compone de 4+1 vistas: Vista Lógica, Vista de Procesos, Vista de Implementación, Vista de Despliegue y estas cuatro regidas por la Vista de Casos de Uso. (33)

Para estructurar el diseño del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas se escogió la arquitectura Modelo Vista Controlador (MVC). La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

En este caso el modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. La vista transforma el modelo en una página Web que permite al usuario interactuar con ella y el controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

3.2.2 Justificación de los Patrones de Diseño utilizados.

Durante el diseño del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas se proponen utilizar los Patrones Generales de Software para Asignación de Responsabilidades (GRASP), que constituyen principios básicos a tener en cuenta cuando se quiere construir eficazmente un software orientado a objetos. Entre los más evidentes en el diseño propuesto se encuentran los patrones: Creador, Experto, Alta Cohesión, Bajo Acoplamiento y Controlador.

- **Creador:** Se asigna la responsabilidad de que una clase B cree un Objeto de la clase A. Las acciones definidas para cada módulo se encuentran en la clase nombremoduloActions. Las acciones contienen toda la lógica de la aplicación. En las acciones se crean los objetos de las clases que representan las entidades y de los objetos de otras clases que intervienen en la lógica del módulo.

- **Experto:** La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. La implementación que realiza Symfony de la arquitectura MVC incluye varias clases, en las cuales se evidencia el uso de este patrón:

  • SfController: Clase perteneciente al controlador que se encarga de decodificar la petición y transferirla a la acción correspondiente.

  • SfRequest: Almacena los elementos que forman la petición (parámetros, cookies, cabeceras, etc.)

  • SfResponse: Contiene las cabeceras de la respuesta y los contenidos. El contenido de este objeto se transforma en la respuesta HTML que se envía al usuario.

- **Alta Cohesión:** Cada elemento del diseño debe realizar una labor única dentro del sistema. La clase nombreActions define las acciones para las plantillas y colabora con otras para realizar diferentes operaciones. Esta clase está formada por diferentes funcionalidades estrechamente relacionadas proporcionándole flexibilidad al software ante grandes cambios.

- **Bajo Acoplamiento:** Debe haber pocas dependencias entre las clases. A cada clase Symfony le asigna una responsabilidad, de forma tal que mantiene pocas dependencias entre las mismas.

- **Controlador:** Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Las peticiones Web son manejadas por el controlador frontal, el cual es el único punto de entrada de toda la aplicación en un determinado entorno. El controlador frontal, al recibir una petición, utiliza el sistema de enrutamiento para enviar la acción al controlador encargado de darle solución.

Para contribuir a una implementación eficiente se hizo necesario el estudio de los patrones del grupo de los cuatro (GOF, Gang Of Four), de ellos se seleccionaron cuatro: Singleton, Abstract Factory, Decorator, Composite.

3.2.3 Modelo de Diseño.

El Modelo de Diseño es un modelo de objetos que describe la realización de los casos de uso y al mismo tiempo constituye una abstracción del modelo de implementación y del código fuente, constituye una entrada esencial a las actividades de implementación y prueba. RUP propone que el Modelo de Diseño básicamente contenga:

- Introducción: Una descripción textual que sirve como breve introducción al modelo.
- Paquetes y Subsistemas de Diseño: Los paquetes y subsistemas del diseño representados en una jerarquía y una breve descripción de ellos.
- Diagramas: los diagramas de clases del diseño y diagramas de interacción (colaboración y/o secuencia) del diseño. Estos últimos también llamados realización de casos de uso.
- Clases, interfaces, relaciones, etc.: contenidas en los paquetes y una breve descripción de ellos. (33)

3.2.3.1 Diagramas de clases del diseño.

Se construyen los diagramas de clases del diseño para representar de forma detallada como se relacionan e interactúan las clases del diseño, ya desde la lógica del programador. A continuación se muestra el diagrama del caso de uso Gestionar encuesta.

Consultar los restantes diagramas de clases del diseño de los casos de uso en el artefacto Modelo de Diseño.

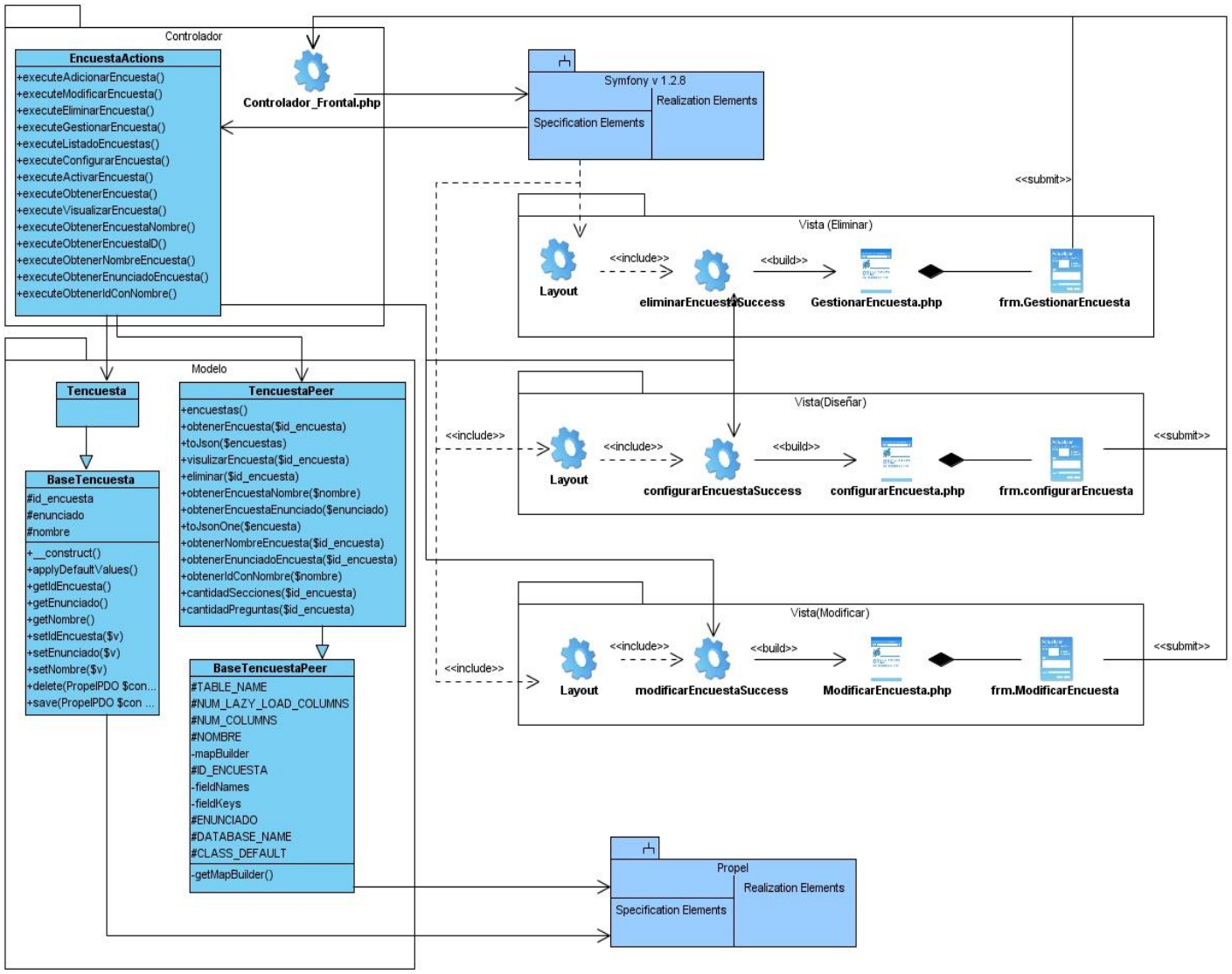


Figura 8: Diagrama de Clases del Diseño. CU Gestionar_encuesta.

La imagen muestra el diagrama de clases del diseño, donde se agrupan en el paquete del modelo las clases pertenecientes a la persistencia de los datos, en el paquete del controlador se encuentran las clases controladoras del sistema y el paquete de la vista contiene las clases necesarias para la construcción de las interfaces, así como los formularios.

Conclusiones

En este capítulo se describió el flujo de trabajo Análisis y Diseño, se mostraron los diagramas de clases tanto del análisis como del diseño, así como el diagrama de colaboración correspondiente a cada caso de uso del sistema, se definió por parte del equipo de arquitectura la utilización de la arquitectura Modelo Vista Controlador para estructurar el diseño del sistema y se definieron un conjunto de patrones de diseño que ayudarán a la posterior implementación del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

CAPÍTULO 4: Validación de los Resultados

Introducción

En el presente capítulo se realizará la validación de los requisitos y el diseño, ya que de esa forma se muestra que los requisitos definen el sistema que realmente el usuario desea y que el diseño que aquí se realizó cumple con los patrones utilizados. Se utilizarán para la validación de los artefactos obtenidos técnicas y métricas.

4.1 Métricas para la Calidad de la Especificación de los Requisitos de Software

Los requisitos, una vez definidos, necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de estos define 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.

Con el objetivo de medir la satisfacción del cliente se aplicaron un conjunto de métricas para la calidad de la Especificación de los Requisitos de Software. A continuación se muestra como se realizó este proceso.

Inicialmente fue necesario calcular el número total de requisitos, para así poder aplicar las métricas que hacen uso de este. (34)

$\mathbf{R_f}$: Número de requisitos funcionales.
$\mathbf{R_{nf}}$: Número de requisitos no funcionales.
$\mathbf{R_t}$: Total de requisitos.

$$\mathbf{R_t = R_f + R_{nf}}$$

$$\mathbf{R_t = 54 + 11}$$

$$\mathbf{R_t = 65}$$

- **Especificidad**

Para determinar la especificidad (ausencia de ambigüedad) de los requisitos se empleó la métrica basada en la consistencia de la interpretación de los revisores para cada requisito. Cuanto más cerca de 1 esté el

valor de $\mathbf{Q_1}$ (grado de especificidad de los requisitos), mayor será la consistencia de la interpretación de los revisores para cada requisito y menor será la ambigüedad de la especificación de los requisitos. (34)

$\mathbf{R_{ii}}$: Número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

$$\mathbf{Q_1} = \frac{\mathbf{R_{ii}}}{\mathbf{R_t}}$$

$$\mathbf{Q_1} = \frac{\mathbf{59}}{\mathbf{65}}$$

$$\mathbf{Q_1} = \mathbf{0.90}$$

- **Corrección**

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 e indica un alto nivel de corrección en la definición de los requisitos. Este valor se calcula como se muestra a continuación: (34)

$\mathbf{Q_2}$: Grado de validación de los requisitos.

$\mathbf{R_c}$: Número de requisitos que se han validado como correctos.

$\mathbf{R_{nv}}$: Número de requisitos que no se han validado como correctos todavía.

$$\mathbf{Q_2} = \frac{\mathbf{R_c}}{\mathbf{R_c} + \mathbf{R_{nv}}}$$

$$\mathbf{Q_2} = \frac{\mathbf{65}}{\mathbf{65} + \mathbf{0}}$$

$$\mathbf{Q_2} = \mathbf{1}$$

- **Compleción**

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 e indica un alto nivel de completitud en la definición de los requisitos. Este valor se calcula como se muestra a continuación: (34)

$\mathbf{n_A}$ **:** Número de requisitos completos.

$\mathbf{n_B}$ **:** Número de requisitos pobremente especificados**.**

$$\mathbf{Q_3 = \frac{n_A}{n_A + n_B}}$$

$$\mathbf{Q_3 = \frac{65}{65 + 0}}$$

$$\mathbf{Q_3 = 1}$$

- **Comprensión**

La comprensión de los requisitos se determinó a partir de la relación que se muestra a continuación. El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1. (34)

$\mathbf{R_{bc}}$ **:** Número de requisitos que todos los revisores entienden.

$$\mathbf{Q_4 = \frac{R_{bc}}{R_t}}$$

$$\mathbf{Q_4 = \frac{65}{65}}$$

$$\mathbf{Q_4 = 1}$$

- **Consistencia interna**

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 y expresa que no existen subconjuntos de requisitos contradictorios. El valor óptimo de esta métrica es el más cercano a 1.

$\mathbf{n_u}$: Número de requisitos especificados.

$\mathbf{n_n}$: Número de requisitos en conflicto con otros requisitos en la especificación.

$$\mathbf{Q_5 = \frac{n_u - n_n}{n_u}}$$

$$\mathbf{Q_5 = \frac{65 - 0}{65}}$$

$$\mathbf{Q_5 = 1}$$

- **Consistencia externa**

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 y expresa que ninguno de los requisitos está en contradicción con lo expresado en documentos de nivel superior, o sea los requisitos del software no pueden contradecir los requisitos del sistema. El valor óptimo de esta métrica es el más cercano a 1. (34)

n_{ec} : Número de requisitos que son consistentes con otros documentos.

$$\mathbf{Q_6} = \frac{\mathbf{n_{ec}}}{\mathbf{R_t}}$$

$$\mathbf{Q_6} = \frac{\mathbf{65}}{\mathbf{65}}$$

$$\mathbf{Q_6} = \mathbf{1}$$

- **Estabilidad**

Para medir la estabilidad de los requisitos de software, en el presente trabajo se aplicó la métrica propia para esto, la cual ofrece valores entre 0 y 1. El mejor valor de $\mathbf{E}$ es el más cercano a 1. La estabilidad de los requisitos se calcula de la siguiente forma:

$\mathbf{E}$: Valor de la estabilidad de los requisitos.

$\mathbf{R_m}$: Número de requisitos modificados, que se obtienen como la sumatoria de los requisitos insertados, modificados y eliminados. (34)

$$\mathbf{E} = \left[\frac{\mathbf{R_t} - \mathbf{R_m}}{\mathbf{R_t}}\right]$$

$$\mathbf{E} = \frac{\mathbf{65} - \mathbf{13}}{\mathbf{65}}$$

$$\mathbf{E} = \mathbf{0.80}$$

Para las métricas especificidad, corrección, comprensión, consistencia interna y externa se recomienda que los resultados estén más cercanos a 1 para ser clasificados como óptimos, el equipo del proyecto llegó a un acuerdo, asignarle la clasificación de satisfactoria a partir de 0.90. El umbral para la métrica de compleción se recomienda que sea un peso aproximado a 0.7, aunque si este es 1 la métrica es clasificada como completa. En el caso de la estabilidad se decidió clasificarla en alta (0.90<=**E**<=1), en media (0.80<=**E**<0.90) y en baja (0.7<=**E**<0.80).

Para un mejor entendimiento de los resultados obtenidos a partir de la aplicación de las métricas, se realizó una gráfica para su ilustración. (Ver anexo 3).

4.2 Validación mediante Prototipos de Interfaz de Usuario no Funcionales

De las técnicas disponibles se decide utilizar el Prototipo de Interfaz de usuario como técnica para validar los requisitos. Su principal objetivo es validar el contenido informativo de las interacciones usuario-sistema descritas en los requisitos definidos en la fase de especificación. Estos prototipos se desarrollan con la herramienta Visual Paradigm definida a utilizar para el desarrollo del software. Estos prototipos son anexados en la Descripción de los Casos de Uso del sistema, con el objetivo de hacer más explícita su descripción.

La validación de los requisitos mediante el prototipo interfaz es sumamente útil porque permite observar las reacciones del usuario, identificar omisiones o malas interpretaciones e incorporar sugerencias e innovaciones antes de entregar una primera versión completa. Los analistas proponen una solución, los usuarios deben entender y validar la propuesta de preferencia antes de que comience el desarrollo propiamente dicho del sistema y con este propósito se construye el prototipo de interfaz.

Una vez aplicada esta técnica se tuvo como resultado una buena aceptación por parte del cliente ya que los requisitos elicitados responden a sus intereses, además demostró que los requisitos no fueron ambiguos e inconsistentes. Esta técnica además permitió mostrar que los errores detectados anteriormente fueron bien corregidos. (35)

4.3 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 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 (35).

Las métricas escogidas para la validación del diseño fueron Métrica de diseño a nivel de componentes (Métrica de cohesión) y Métrica orientada a clases (Tamaño de clase).

4.3.1 Métrica de diseño a nivel de componentes

- **Métrica de cohesión**

Lograr una alta cohesión constituye una buena práctica para el diseño, como resultado de aplicar los patrones GRASP en la construcción del mismo. Para medir el índice de cohesión de los elementos que conforman el diseño de los módulos del sistema se aplicó la métrica de Bieman y Ott (35).

Para ello se analizaron elementos pertenecientes a los principales conceptos que plantea la métrica, dígase: Porción de datos, Símbolos léxicos (tokens) de datos, Señales de unión, Señales de súper-unión y Cohesión. Para proceder al análisis de cada uno de estos conceptos se recogieron en una tabla los datos que ilustran el nivel de uso de las principales clases diseñadas para los módulos del sistema. **Ver artefacto Modelo de validación.**

La cohesión funcional se determina con dos enfoques:

1. Determinando la cohesión funcional fuerte (CFF) y la pegajosidad: se obtienen cuando el resultado de la métrica es de 1.

- Se define como: CFF **= número de súper adhesivos (i) / número de elementos (i)**

2. . Determinando la cohesión funcional débil (CFD):

- Se define como: CFD = **número de adhesivos (i)/ número de elementos (i)**

Adhesivo: Se le llamará adhesivo a un elemento que aparece en dos o más rebanadas.

Súper adhesivo: Se denomina súper adhesivo a un elemento que está en todos los elementos de un módulo.

(i): Se define como la muestra.

Según los datos de las clases analizadas se tiene que:

número de elementos (i) = 53

número de súper adhesivos (i) = 0

número de adhesivos (i) = 45

CFD = número de adhesivos (i) / número de elementos (i)

$$\mathbf{CFD} = \frac{\mathbf{45}}{\mathbf{53}}$$

$$\mathbf{CFD} = \mathbf{0.84}$$

CFF = número de súper adhesivos (i) / número de elementos (i)

$$\mathbf{CFF} = \frac{\mathbf{0}}{\mathbf{53}}$$

$$\mathbf{CFF} = \mathbf{0}$$

La métrica de Bieman y Ott plantea que mientras más cerca están los valores de CFF y CFD de 1 mayor será la cohesión de los módulos. Los resultados demuestran que no hay una cohesión funcional fuerte, pero la relación del número de clases adhesivas con el número total de elementos de la muestra, determinados por la CFD = 0.84, está cercana a 1, lo que demuestra que el diseño de las clases de los módulos poseen una cohesión funcional con un 84% de fortaleza.

4.3.2 Métricas orientadas a clases. Tamaño de clase (TC)

El tamaño general de una clase se puede determinar siguiendo los planteamientos a continuación:

- El número total de operaciones (tanto operaciones heredadas como operaciones privadas de la instancia) que están encapsuladas dentro de la clase.
- El número de atributos (tanto atributos heredados como atributos privados de la instancia) que están encapsulados en la clase.

Si existen valores grandes de TC éstos estarán demostrando que una clase puede tener demasiada responsabilidad, lo cual reduciría la reutilización de la clase y hará complicada la implementación y la prueba. De forma contraria sucede si los valores TC son de menor valor. Por otra parte es necesaria una evaluación concreta de las métricas mediante los umbrales. Finalmente se calcula los promedios correspondientes a los diferentes valores para tener una estimación general del sistema (35). Con el objetivo de comprobar el adecuado diseño de las clases y el nivel de reutilización de las mismas se aplicó la métrica del TC. Se aplicará esta métrica para las principales clases de los módulos definidos en el sistema.

Para proceder al análisis de esta métrica se recogieron en una tabla los datos Nombre de la Clase, Número de Atributos y Número de Operaciones, estos dos últimos ilustran la cantidad de cada uno de ellos en cada clase. **Ver artefacto Modelo de validación.**

Se presentó un **total de 53 clases** para un **promedio de atributos de 5.18** y **un promedio de operaciones de 10.11**.

De esta forma el umbral queda con los datos mostrados a continuación:

Umbral	Tamaño	Cantidad de clases
<=20	Pequeño	49
>20<=30	Medio	2
>30	Grande	2

Tabla 3 Umbral de las clases.

Para un mejor entendimiento de los resultados obtenidos a partir de la aplicación de esta métrica, se realizó una gráfica para su ilustración.

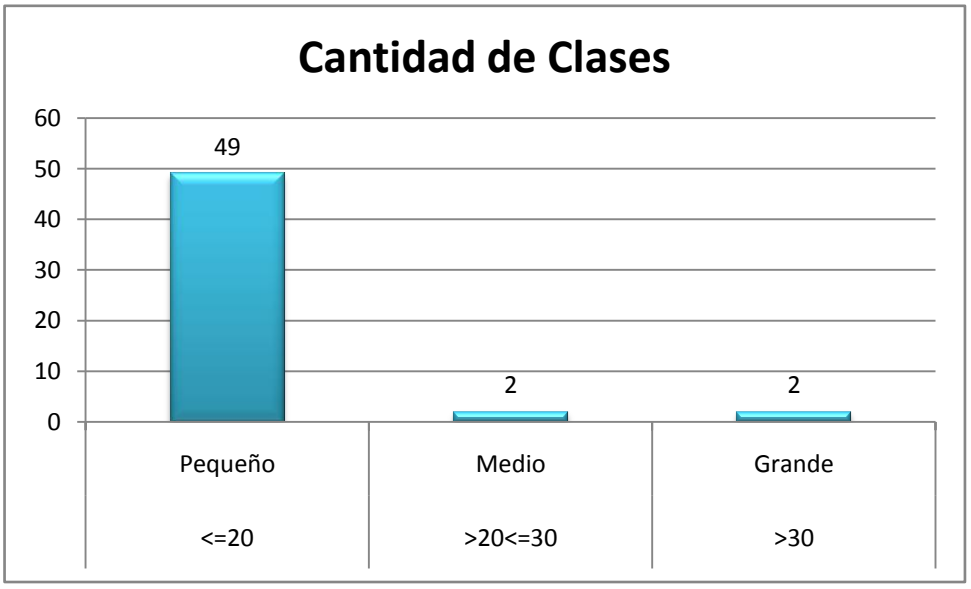


Figura 9: Número de clases por categorías.

El 92 % de las clases diseñadas están consideradas como pequeñas, lo que facilitará el proceso de construcción de los módulos del sistema.

Para la validación de los artefactos obtenidos se le realizó revisiones por parte del Grupo de Calidad Centro CEGEL. En estas se obtuvieron 2 no conformidades, las cuales fueron corregidas, quedando liberados los artefactos realizados. (Ver anexo 5).

Conclusiones

En este capítulo se analizaron los resultados obtenidos durante el desarrollo del trabajo, es decir, se avaluaron todos los artefactos obtenidos, arribándose a las siguientes conclusiones:

- La aplicación de las métricas para la calidad de la funcionalidad del Documento de ER demostró que estas especificaciones se corroboraron con las necesidades de clientes/usuarios finales y que son correctas las interpretaciones por parte del equipo de desarrollo de Software.
- Se mostró de una forma clara como debe ser el sistema una vez terminado a través de los prototipos de interfaz de usuario diseñados para que el cliente tenga noción de su producto final y muestre su criterio.
- Las métricas aplicadas a los elementos del diseño de los módulos validan la calidad del diseño elaborado.

Conclusiones

Durante el desarrollo de este trabajo se expuso la necesidad de desarrollar el Análisis y Diseño del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas con el fin de darle respuesta a la problemática actualmente surgida en la Universidad de las Ciencias Informáticas con respecto al aprendizaje autónomo de idiomas extranjeros de los estudiantes. De esta forma surge el proyecto de Innovación Pedagógica: Metodología de Evaluación para Aprendizaje Autónomo de Idiomas con el objetivo de brindarle servicios a toda la comunidad universitaria, además de permitir un ambiente alternativo de desarrollo, donde el estudiante sea el encargado de trabajar de manera independiente, profundizando en sus debilidades y en los temas que son de su interés, además de evaluar de manera efectiva el desarrollo de las habilidades comunicativas de forma autónoma.

Luego de realizar todo un estudio referente a lo antes mencionado se arribó a las siguientes conclusiones:

- Se cumplió el objetivo general trazado para este Trabajo de Diploma: Realizar el Análisis y Diseño del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.
- Se realizó una amplia investigación de las tendencias y tecnologías actuales, permitiendo seleccionar las herramientas adecuadas para el desarrollo de la solución propuesta.
- El levantamiento de requisitos permitió identificar las funcionalidades y cualidades que debe tener el sistema, conformado por 65 requisitos, 54 funcionales y 11 no funcionales, identificando los casos de uso y obteniendo el diagrama de casos de uso del sistema.
- Se desarrolló el análisis y diseño del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas, modelando los diagramas de clases del análisis y sus realizaciones, además de los diagramas de clases del diseño utilizando para estos los patrones GRASP y GOF.
- La aplicación del prototipado de interfaces, además de las métricas y técnicas propuestas por varios autores posibilitó validar los resultados obtenidos durante el análisis y diseño, obteniéndose una cohesión funcional con un 84% de fortaleza y el 92% de las clases fueron clasificas como pequeñas.

Recomendaciones

A lo largo de este trabajo, se pudo apreciar cómo se cumplió con cada uno de los objetivos trazados en el mismo, no obstante, se realizan varias recomendaciones a aquellos que darán continuación a dicho trabajo, entre las que se encuentran las siguientes:

- Elaborar los artefactos restantes pertenecientes al diseño, que son necesarios para continuar con el desarrollo del sistema.
- Continuar perfeccionando el modelado de sistema mediante la actualización de los cambios que sean necesarios durante las etapas de diseño, implementación y pruebas.
- Profundizar en el estudio de la aplicación de la Ingeniería de Requisitos aplicando la actividad de Administración de Requisitos.

Referencias Bibliográficas

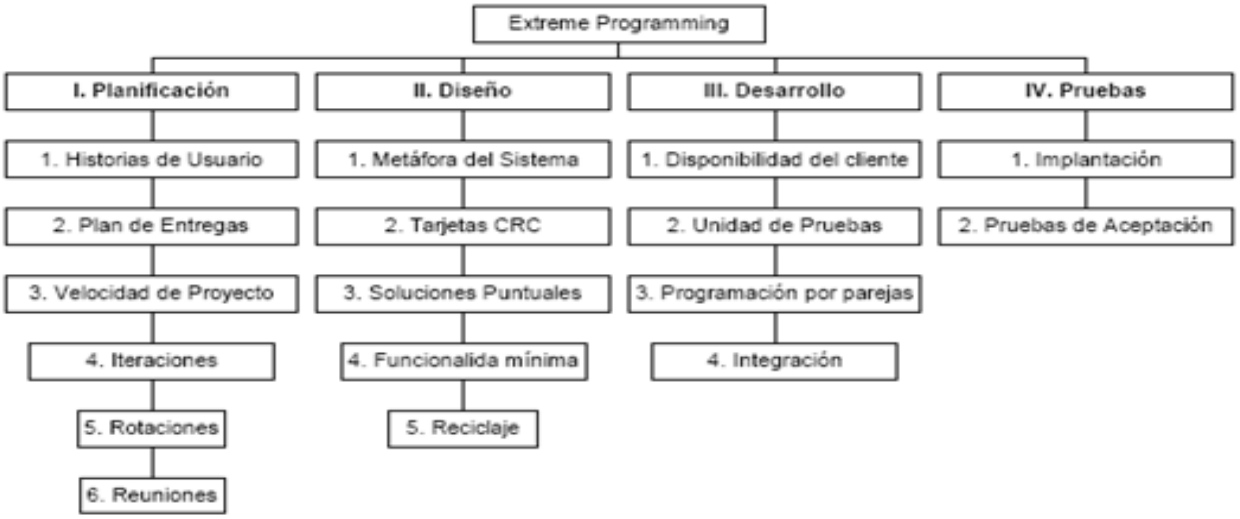
1- Alfin EEES, 2009. Disponible en: http://www.mariapinto.es/alfineees/autonomo/que.htm
2- Castillo González, Reinier y de la Rosa Rodríguez, José M. Módulo Gestión del Aprendizaje del Centro Virtual de Autoaprendizaje de lenguas Extranjeras. Tesis (Ingeniero en Ciencias Informáticas), Ciudad de La Habana, Cuba: UCI, Facultad 9, 2008. 8 p.
3- Villanueva, Mª Luisa. Los Estilos de aprendizaje de Lenguas. (1997), nº.
4- Infojobs.net Disponible en: http://www.conocimientosweb.net/zip/article108.html
5- Martínez Márquez, Yoan. Propuesta de sistema de evaluación del aprendizaje autónomo del idioma inglés en un entorno virtual de aprendizaje en la Universidad de las Ciencias Informáticas. Tesis (Máster en Tecnologías en los procesos Educativos), Ciudad de La Habana, Cuba: UCI, 2007.
6- Giraffa, L. M. M. Selección y adopción de estrategias de educación en Sistemas Tutores Inteligentes. Porto Alegre: CPGCC/UFRGS. (1997).
7- López Barrio, C. Metodología de Desarrollo: Programación Extrema. Tesis (Doctorado en Ingeniería de Sistemas Electrónicos para Entornos Inteligentes), Dpto. Ingeniería Electrónica - ETSIT-UPM, [25/11/2005]. Disponible en: http://www-lsi.die.upm.es/~carreras/ISSE/programacion_extrema_1.x2.pdf
8- Cleger Despaigne, Eliober y Tornés Montes de Oca, Annarella María. Análisis y Diseño de una herramienta interactiva de simulación de procesos: Nodo Virtual de Procesos. Segunda Iteración. Tesis (ingeniero en ciencias informáticas). Ciudad de la Habana, UCI facultad 3, Abril 2009.
9- Ed. Addison Wesley. Extreme Programming Explained. Embrace Change. Notas sobre Metodologías Ágiles. 02 de Abril de 2008. Disponible en: http://migueljaque.com/index.php/metodologias/xp/29-xp/63-programacionextrema
10- González Barbone, Víctor A. XP: Extreme Programming (Programación Extrema). Instituto de Ingeniería Eléctrica. Facultad de Ingeniería. Montevideo, Uruguay. Disponible en: http://iie.fing.edu.uy/~nacho/blandos/seminario/XProg1.html
11- Blog de WordPress.com, 14 de Febrero, 2008, Disponible en: http://arturoweb.wordpress.com/2008/02/14/scrum-metodologia-agil-de-desarrollo/

12- debug_mode=ON, 2009, Disponible en: http://es.debugmodeon.com/articulo/scrum-una-metodologia-agil-i
13- Itzcoalt Álvarez M. Desarrollo Ágil con SCRUM [en línea], Disponible en: http://www.sg.com.mx/sg07/presentaciones/Mejora%20de%20procesos/SG07.P02.Scrum.pdf
14- Rueda Chacón, Julio César. Aplicación de la Metodología RUP para el Desarrollo Rápido de Aplicaciones Basado en el Estándar J2EE. Tesis (Ingeniero en Ciencias y Sistemas). Guatemala: Universidad de San Carlos de Guatemala. Facultad de Ingeniería. Marzo de 2006. Disponible en: http://biblioteca.usac.edu.gt/tesis/08/08_7691.pdf
15- Scribd, 2009, Disponible en: http://www.scribd.com/doc/3708746/Aprendiendo-UML-en-24-horas
16- Mora, Beatriz; Ruiz, Francisco; García, Félix y Piattini, Mario. Experiencia en transformación de modelos de procesos de negocios desde BPMN a XPDL [en línea]. España, Universidad de Castilla-La Mancha, Escuela Superior de Informática [fecha consulta 23-02-2010] Disponible en: http://kuainasi.ciens.ucv.ve/ideas07/documentos/articulos_ideas/Articulo35.pdf
17- BizAgi. Disponible en: http://www.bizagi.com/docs/BPMNbyExampleSPA.pdf
18- INTALIO (BPM + BPMN + BPEL + OPEN SOURCE), Cejas, Julio. 21 de febrero de 2010. Disponible en: http://intaliobpm.blogspot.com/
19- Instituto Nacional de Estadística informática, Herramientas Case[en línea].Talleres de la Oficina de Impresiones de la Oficina Técnica de Difusión Estadística y Tecnología Informática del Instituto Nacional de Estadística e Informática (INEI), colección cultura informática. Noviembre de 1999. [fecha de consulta:19 Marzo 2010]. Disponible en: http://jhoel-jp.tripod.com/paginas/herramientascase.pdf ISSN 875-99-OI-OTDETI-INEI
20- Sitio de Descargas de Software, Marzo 05, 2007. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/
21- CEDS e Learning. Disponible en: http://ceds.nauta.es/informes/case04.htm
22- Boost Productivity with Innovative and Intuitive Technologies. Disponible en: http://www.visual-paradigm.com/aboutus/10reasons.jsp
23- Grupo Soluciones GSINNOVA, Disponible en: http://www.rational.com.ar/herramientas/roseenterprise.html

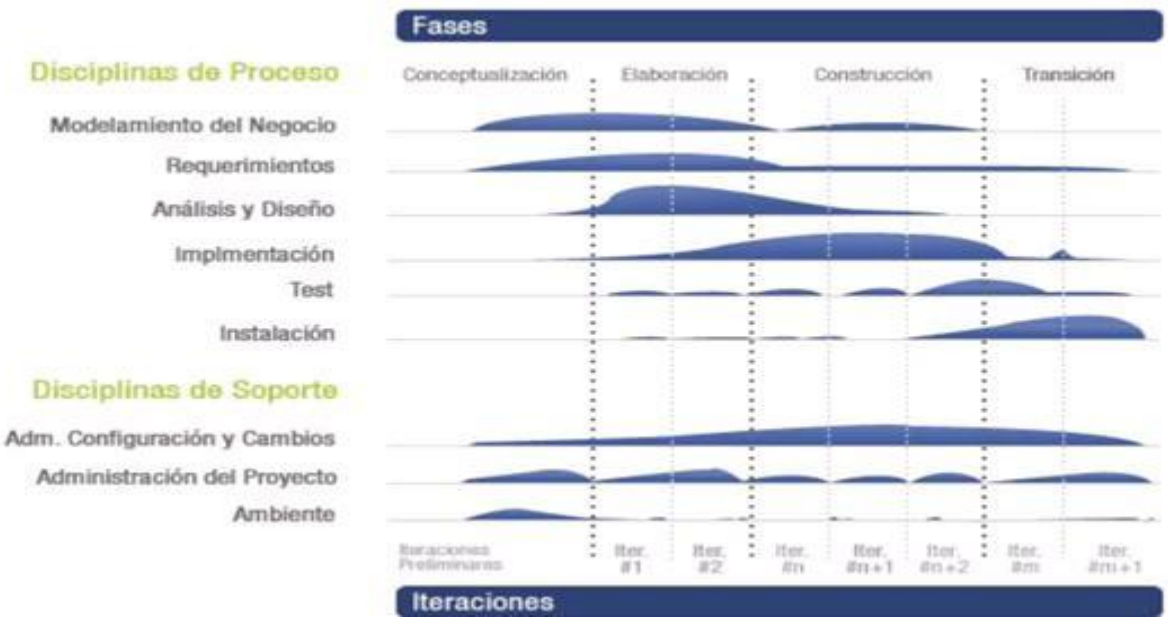
24- Benítez Morales, Erick Alexander. Herramienta case "ArgoUML"[en línea]. Cuidadela Don Bosco: Universidad Don Bosco Facultad de Ingeniería, 2006[fecha de consulta: 23 febrero 2010]. Disponible en: http://erickbenitez.iespana.es/Herramientas%20case.pdf
25- PHP experto. Jaime M. Tan Nozawa.9 de diciembre de 2008. Disponible en: http://phpexperto.blogspot.com/2008/12/comendo-con-uml-y-php.html
26- Java y sus ventajas. Disponible en: http://e-articles.info/t/i/5006/l/es/
27- ZatorSystems, 2008. Disponible en: http://www.zator.com/Cpp/E1_2.htm
28- Maestros de web, 23 de mayo, 2001. Disponible en: http://www.maestrosdelweb.com/editorial/phpintro/
29- Programación en Castellano, 1997. Disponible en: http://www.programacion.com/php/articulo/porquephp/
30- Potencier, Fabien y Zaninotto, Francois. Symfony la guía definitiva. Apress. 21 de octubre de 2007. ISBN-13:978-1590597866.
31- PostGreSQL vs. MySQL. Disponible en: http://www.netpecos.org/docs/mysql_postgres/x57.html
32- PostGreSQL vs. MySQL. Disponible en: http://www.netpecos.org/docs/mysql_postgres/x15.html
33- Jacobson, I, Booch, G y Rumbaugh, J. El Proceso Unificado de Desarrollo de Software. La Habana: Félix Varela, 2004.
34- Davis, S. Overmyer, y otros. 1993. Identifying and measuring quality in software requirements specification. California: Los Alamitos, 1993.
35- Roger S. Pressman. Ingeniería de software un enfoque práctico. Quinta edición.1998.

Anexos

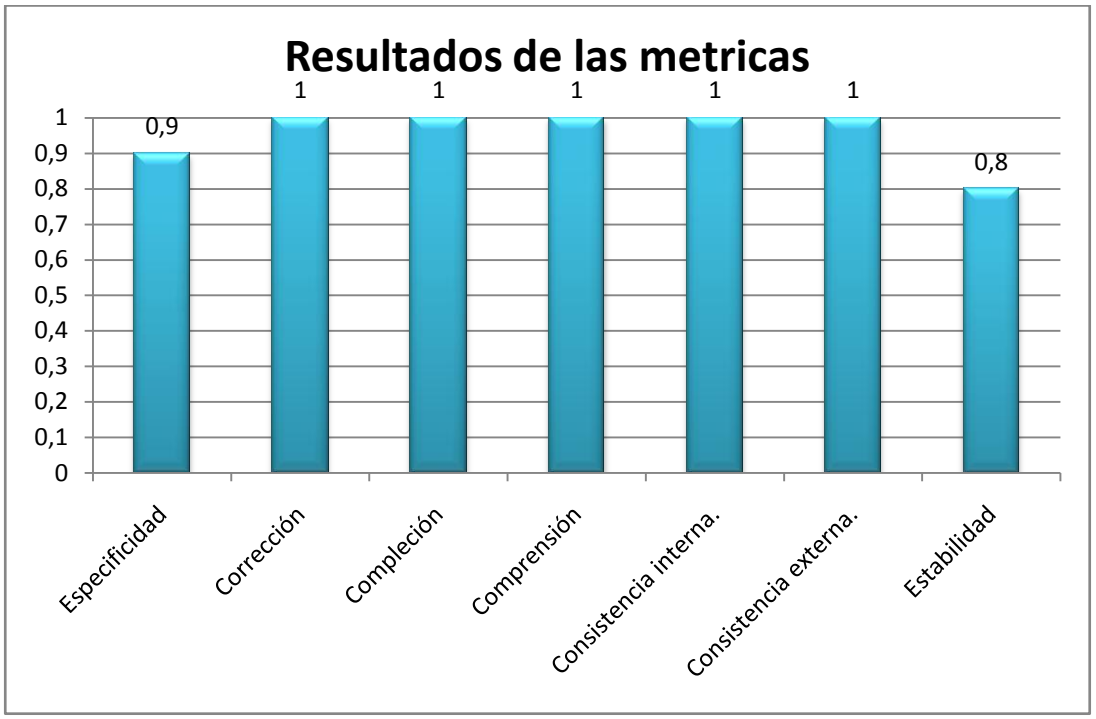
Anexo 1: Fases de XP.



Anexo 2: Dimensiones de RUP.



Anexo 3: Resultados de las métricas aplicadas a la Especificación de requisitos.



Anexo 4: Carta de Liberación de Artefactos.

UCI
Universidad de las Ciencias Informáticas

Acta de Liberación de Artefactos, Grupo de Calidad Centro CEGEL de la Facultad 15 de la Universidad de las Ciencias Informáticas.

Miércoles, 26 de mayo de 2010.

Luego de haber efectuado 2 iteraciones de revisiones a los artefactos: Especificación de Requisitos, Modelo de Sistema, Modelo de diseño y Modelo de Dominio del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas del proyecto VIRTEVAL del Centro CEGEL de la Facultad 15 y haberse detectado un promedio de 5 No Conformidades, se puede afirmar que se han corregido los defectos encontrados, por lo que quedan liberados los artefactos.

Firma del Asesor y Jefe del Grupo de Calidad Centro CEGEL

Ing. Raúl Velázquez Alvarèz

Decanato Facultad