Universidad de las Ciencias Informáticas

Facultad 3

Herramienta Informática para la evaluación de trabajos de curso mediante la computación con palabras

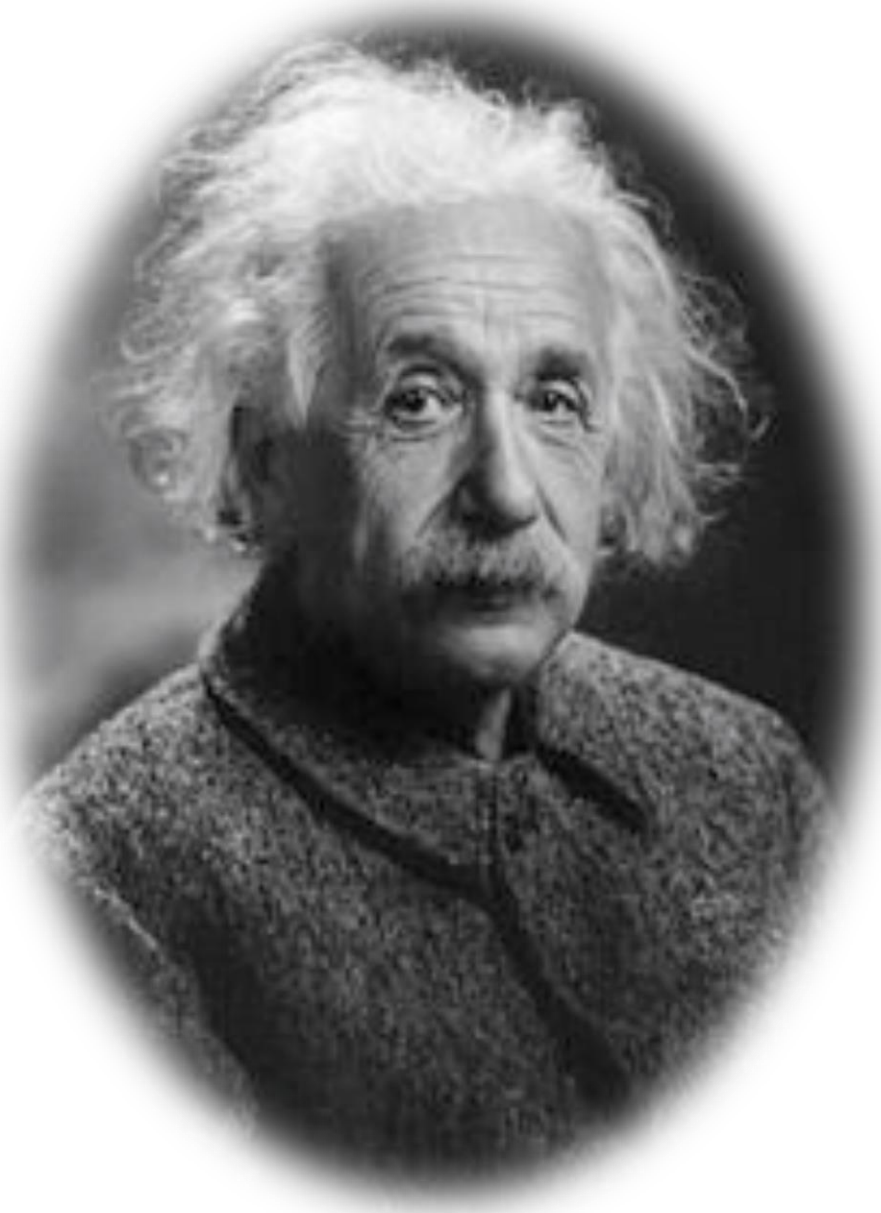
Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Jenniffer Maulini Ruiz
Ingrid García Ponce

Tutores: Dra. C Marieta Peña Abreu,
MSc. Carlos Rafael Rodríguez Rodríguez
Ing. Yadira García

La Habana, junio de 2018

"Año 60 de la Revolución



"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber".

Albert Einstein

DECLARACIÒN DE AUTORÌA

DECLARACIÓN DE AUTORÍA

Declaramos ser autoras de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Jenniffer Maulini Ruiz

_____

Firma del Autor

Ingrid García Ponce

_____

Firma del Autor

Dra. C. Marieta Peña Abreu

_____

Firma del Tutor

Ing. Yadira García

_____

Firma del Tutor

MSc. Carlos Rafael Rodríguez Rodríguez

_____

Firma del Tutor

DATOS DEL CONTACTO

DATOS DE CONTACTO

Autor: Jenniffer Maulini Ruiz

Correo electrónico: jmaulini@estudiantes.uci.cu

Autor: Ingrid García Ponce

Correo electrónico: igponce@estudiantes.uci.cu

Tutor: Marieta Peña Abreu

Correo electrónico: mpabreu@uci.cu

Tutor: Carlos Rafael Rodríguez Rodríguez

Correo electrónico: rodriguezr@uci.cu

Tutor: Yadira García

Correo electrónico: yggarcia@uci.cu

Agradecimientos

A la revolución y a Fidel por haber creado una universidad con tantos recursos y comodidades en la cual me pude forjar como ingeniera. Aunque para muchos sea una bobería le agradezco a mis 6 años de carrera, en ella he vivido momentos inolvidables. Que me ha permitido conocer y estar en momentos y lugares que a lo mejor nunca hubiese imaginado esta

A mis formadores, personas de gran sabiduría quienes se han esforzado por ayudarme a llegar al punto en el que me encuentro, sencillo no ha sido el proceso, pero gracias a las ganas de transmitirme sus conocimientos y dedicación, gracias a ustedes he logrado importantes objetivos como culminar el desarrollo de mi tesis con éxito y obtener una titulación

A mis tutores por el apoyo brindado, y por confiar en nosotras para brindarnos un tema el cual nos ayude a llegar al título gracias a mi tribunal por los señalamientos realizados, aunque pensábamos que no… esos señalamientos son los que nos hacen mejores personas y profesionales.

A lo largo de esta carrera me he encontrado con personas las cuales sin su ayuda tal vez no hubiese llegado aquí personas que me han brindado su mano, su pecho y su aliento

A mis amigos de la universidad...gracias a ustedes por hacer el trance más divertido

A Mireya...por todas sus locuras...por sus videos de risa… aunque al principio me estresabas...sabes que te quiero aunque no sea la madrina del bebé..y que siempre voy a recordarte como aquella niña con problemas que llego el primer día al 57 .... te quiero y lo sabes...

Roniel...gracias por tus consejos...al final eres mi esposo, por tus gritos y regaños cuando hacía algo malo...pero siempre cuidándome...te quiero negro

A la hermandad por tantas noches de fiesta...son únicos.

A Roly por tantas noches de apoyos y tantas palabras de aliento.

A mis compañeros de aula, porque me daban animo cada día para tener ganas de ir.

A mi grupito de estudio de las noches Anayanci, Rosalia, Julio, Roniel.

A mi compañera de tesis por aguantarme, creo que cuando te escogí fue una de las mejores decisiones que he tomado en mi vida.

A mi mejor amiga Dayana gracias por estos últimos años de apoyo, te quiero mucho y te agradezco tanto apoyo a veces sin merecerlo.

A mi familia

A mis primos Leysa, Rubencito hay primo que decirte a ti...nadie sabe mejor que nosotros de nuestra amistad ...de nuestro pacto...como dice mi tío cuando a Rube le pasa algo la única que lo sabe es Yeni...primo tú has sido mi héroe y te lo dije hace 2 años...esa fuerza que sacas para enfrentar los problemas y seguir con tus metas...no me creo capaz de hacerlo...tal vez nadie te entienda y te reprochen, pero yo no...porque yo si hubiese perdido a mi madre días antes de graduarme yo no hubiese llegado ahí.

AGRADECIMIENTOS

aunque fuera ese su mayor deseo...por eso es que parte de este título va a ti...porque casi al final me vencía, pero tú fuiste mi luz...te amooo

A mis tíos: Ale, Alina, Ruben, Martica

Aunque del cuarteto falte una...no importa para mi siguen siendo 4...en mi corazón son 4...los quiero a los 4 por igual…y les agradezco que desde niña cargaban conmigo a todos lados como si fuera de ustedes una más...siempre me tenían presente y me expresaban su cariño como si fuera de su propio ser.

A mis hermanos Nhaty y Eduard

A mis abuelos Miriam, Eugenio, Emilio

A mi mama: este título es tuyo completo...cuando me lo entreguen te lo regalo…recuerdo cuando entre a la universidad que no quería seguir...inclusive ni iba al aula... y por eso suspendí mi primer año...te dije no quiero seguir...y me enseñaste y me dijiste si yo supiera que no dabas yo misma te saco de la carrera...

pero como es malacrianza tu va a seguir y vas a terminar...pos y ves aquí estoy...te agradezco todo lo que me has dado. y lo que no también...te agradezco que en mis momentos más difíciles a tu manera has sabido guiarme. Has sabido enseñarme y aunque tuvieras un dolor no me lo has demostrado...gracias por todo, pero sobre todo gracias por al final decidir tenerme. Te amo

Carlo

papi muchas gracias porque me cuidaste desde los 4 años. Y aunque no tengas hijos supiste quererme como una. Sabes que siempre vamos a estar juntos en las buenas y las malas. Y te agradezco un montón por cuidar de mi mama...en mis 15 fuiste mi galán. En mi tesis mi papa y espero que en mi boda el que me lleve al altar te amo...

Jenniffer Maulini Ruiz

Agradecimientos

Quiero agradecer a mis padres por su apoyo incondicional, por estar a mi lado en todo momento, por aconsejarme siempre, aunque no les haga caso casi nunca. Por ser los mejores padres del mundo.

A mi hermanita bella que siempre está ahí cuando lo necesito y a pesar de ser una niña me ha demostrado que puedo contar con ella siempre.

A mi familia en general, que siempre me han apoyado cuando más o he necesitado sin pedir nada a cambio.

A mis amistades del barrio por ser tan geniales e incondicionales conmigo.

A mi Pai, no sé qué hubiera sido de mí en esta escuela sin tenerte a mi lado dándome consejos y pasando momentos inolvidables, por portarte como un hermano más que como amigo. Muchas gracias.

A mis niñas Nailee y Claudia por estar siempre a mi lado cuando más las he necesitado, por todos los locos e inolvidables momentos que hemos pasado juntas…y los que faltan.

A mis pupusas, por hacerme pasar los mejores años de la uci.

A mi turra Yanet, que a pesar de estar lejos siempre la tengo presente. Por estar conmigo en todo momento y ser tan buena amiga, por eso y mucho más… gracias.

A mi novio, por ser tan especial, por apoyarme siempre, por estar conmigo en las buenas y en las malas. Por todos los momentos tan lindos que hemos vivido juntos, y los que nos quedan que serán muchos. No me alcanzan las palabras para agradecerte todo los has hecho por mí.

A mi compañera de tesis, por no darse nunca por vencida, aunque no fueron tiempos para nada fáciles. Por apoyarme siempre y ayudarme a lograr este título que tanto nos merecemos.

A Juan Carlos, flaqui, no sé qué hubiera sido de nosotras sin ti…te debemos más de lo que te imaginas. Por eso y todos los momentos lindos y divertidos que hemos pasado juntos. muchas gracias.

A mis amigos del baile … definitivamente esa etapa de mi vida fue muy importante, gracias por vivirla conmigo, por todas las risas, los gritos, las locuras, por todo muchas gracias.

A mis compañeros del 3502 que hicieron que estos 5 años en la universidad fueran tan especiales.

A todos los que de una forma u otra hicieron que estos años en la universidad fueran los mejores. Muchas gracias.

Ingrid García Ponce

DEDICATORIA

A mis padres por
apoyarme en todo
momento.
A mi hermana, por estar
siempre a mi lado y
brindarme su apoyo
incondicional.
Ingrid García Ponce.

A mi madre por ser una mujer que simplemente me hace llenar de orgullo, te amo y no va haber manera de devolverte tanto que me has ofrecido desde que incluso no hubiera nacido.
Esta tesis es un logro más que llevo a cabo, y sin lugar a dudas ha sido en gran parte a ti, no se en donde me encontraría de no ser por tu ayuda, tu compañía y amor.
A mí, por cada día demostrarme a mí misma el simple hecho de superación, de que, si se puede y que, aunque no lo crea puedo lograr muchas cosas con un poco de presión.
Jenniffer Mauilini Ruiz

RESUMEN

La evaluación del aprendizaje tiene un rol fundamental en el desarrollo del proceso docente educativo ya que comprueba el grado de cumplimiento de los objetivos mediante la valoración de conocimientos y habilidades. Contribuye, además desde la acción educativa a la formación de valores por lo cual es esencial realizarla de manera dinámica. En la actualidad, a los profesores universitarios les resulta de vital importancia establecer una nota flexible a los estudiantes en sus trabajos de curso, de esta forma se puede diferenciar los resultados de los mismos según sus conocimientos y habilidades adquiridas a lo largo de la asignatura. Esta necesidad ha traído consigo diversas investigaciones y propuestas para agilizar y mejorar el proceso de evaluación. Este trabajo propone una herramienta capaz de ayudar al profesor a evaluar de manera integral el aprendizaje en los trabajos de curso de la disciplina Ingeniería y Gestión de Software. La evaluación de los trabajos de curso de manera determinista puede provocar variabilidad en los datos, imprecisión y vaguedad en la información, dado que el mismo es llevado a cabo por personas. La propuesta realizada permite la evaluación por parte de múltiples profesores utilizando múltiples criterios, facilitando la toma de decisiones sobre la evaluación integrada del estudiante. Los trabajos de cursos se evalúan utilizando el modelo de representación lingüística 2-tuplas. La utilización de este modelo permite expresar las valoraciones de los profesores en tres dominios: numérico, intervalar y lingüístico, sin pérdida de información en el proceso evaluativo teniendo en cuenta las evaluaciones propuestas en el reglamento docente.

Palabras claves: computación con palabras, evaluación.

ÌNDICE

ÍNDICE
Índice de tablas XIII
Índice de imágenes 14
INTRODUCCIÓN 1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA ..... - 5 -
1.1 Introducción ..... - 5 -
1.2 Principales conceptos ..... - 5 -
1.3 Métodos para la evaluación. ..... - 5 -
1.4 Método para la evaluación del aprendizaje en los proyectos de curso, utilizando Computación con palabras. ..... - 2 -
1.5 Técnicas de Soft Computing ..... - 4 -
1.5.1 Computación con palabras. ..... - 5 -
1.6 Herramientas informáticas para la evaluación. ..... - 6 -
1.6.1 Valoración de las herramientas ..... - 7 -
1.7 Metodología de desarrollo ..... - 7 -
1.8 Herramientas, tecnologías y lenguajes ..... - 9 -
1.8.1 Lenguaje de modelado ..... - 9 -
1.8.2 Herramienta de modelado ..... - 9 -
1.8.3 Lenguaje de programación ..... - 9 -
1.8.4 Entorno de Desarrollo Integrado ..... - 10 -
1.8.5 Sistema Gestor de Base de Datos (SGBD) ..... - 11 -
1.8.6 Servidor web ..... - 11 -
1.9 Patrones para el desarrollo de software ..... - 12 -
1.9.1 Patrón arquitectónico ..... - 12 -
1.9.2 Patrones de diseño ..... - 12 -
1.10 Métricas ..... - 13 -
1.10.1 Métricas para diseño ..... - 14 -
1.11 Evaluación de software ..... - 16 -
1.11.1 Pruebas de caja blanca ..... - 16 -
1.11.2 Pruebas de caja negra ..... - 17 -
1.12 Conclusiones parciales ..... - 17 -

ÌNDICE

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN ..... - 18 -
Introducción ..... - 18 -
Descripción general de la propuesta de solución ..... - 18 -
2.1 Roles relacionados con el sistema ..... - 18 -
Modelo conceptual ..... - 18 -
2.2 Fase de planificación ..... - 19 -
2.2.1 Requisitos del software ..... - 20 -
2.2.1.1 Requisitos funcionales ..... - 20 -
2.2.1.2 Requisitos no funcionales ..... - 23 -
2.2.2 Historias de usuario ..... - 25 -
2.2.3 Plan de iteraciones ..... - 25 -
2.3 Fase de diseño ..... - 28 -
2.3.1 Tarjetas clase-responsabilidad-colaborador (CRC) ..... - 28 -
2.3.2 Modelo de datos ..... - 29 -
2.3.3 Arquitectura del sistema ..... - 30 -
2.4 Patrones de diseño ..... - 31 -
2.4.1 Patrones generales de software para asignación de responsabilidades (GRASP) ... - 31 -
2.4.2 Patrones GoF aplicados ..... - 33 -
2.5 Validación del diseño ..... - 34 -
2.5.1 Tamaño Operacional de Clases (TOC) ..... - 34 -
2.5.2 Relaciones entre Clases (RC) ..... - 35 -
2.6 Conclusiones parciales ..... - 36 -
CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA ..... - 37 -
3.1 Introducción ..... - 37 -
3.2 Fase de desarrollo ..... - 37 -
3.2.1 Tareas de ingeniería por iteraciones ..... - 37 -
3.2.2 Tareas detalladas ..... - 43 -
3.2.3 Estándares de codificación ..... - 43 -
3.3 Fase de pruebas ..... - 44 -
3.3.1 Pruebas de caja blanca ..... - 44 -

ÌNDICE

3.3.2 Pruebas de caja negra. ..... - 47 -

3.3.3 Caso de estudio. ..... - 52 -

3.3.4 Pruebas de aceptación ..... - 56 -

3.4 Conclusiones parciales: ..... - 60 -

CONCLUSIONES ..... - 61 -

RECOMENDACIONES ..... - 62 -

BIBLIOGRAFÍA ..... - 63 -

ANEXOS ..... **¡Error! Marcador no definido.**

Anexo 1. Entrevista realizada a profesores del departamento de Ingeniería y Gestión de Software, facultad 3. **¡Error! Marcador no definido.**

Anexo 2. Certificado de Aceptación del Producto. ..... **¡Error! Marcador no definido.**

Anexo 3. Historias de usuario ..... **¡Error! Marcador no definido.**

Anexo 4. Tarjetas Clase-Responsabilidad-Colaborador. ..... **¡Error! Marcador no definido.**

Anexo 5. Tareas de ingeniería detalladas. ..... **¡Error! Marcador no definido.**

ÌNDICE DE TABLAS

Índice de tablas

Tabla 1. Comparación de los modelos lingüísticos. (Fuente: Elaboración propia) ..... - 6 -
Tabla 2.Comparación de las herramientas de evaluación de trabajos estudiadas. (Fuene: Elaboración propia) ... - 7 -
Tabla 3. Criterios de evaluación para la métrica TOC. (Fuente: Elaboración propia) ..... - 14 -
Tabla 4.Criterios de evaluación para la métrica RC. (Fuete: Elaboración propia)..... - 15 -
Tabla 5. Roles relacionados con el sistema. (Fuente: Elaboración propia)..... - 18 -
Tabla 6. Requisitos funcionales. (Fuente: Elaboración propia)..... - 20 -
Tabla 7. Requisitos no funcionales. (Fuente: Elaboración propia)..... - 23 -
Tabla 8. Plan de iteraciones. (Fuente: Elaboración propia) ..... - 26 -
Tabla 9. Tarjeta CRC de la clase Proyecto. (Fuente: Elaboración propia) ..... - 29 -
Tabla 10. Tareas ingenieriles por iteraciones. (Fuente: Elaboración propia) ..... - 37 -
Tabla 11.Tarea de ingeniería de la historia de usuario: "Adicionar Proyecto de Curso". (Fuente: Elaboración propia) ..... - 43 -
Tabla 12. Caso de prueba Camino 1. (Fuente: Elaboración propia)..... - 46 -
Tabla 133. Caso de prueba Camino 1. (Fuente: Elaboración propia)..... - 46 -
Tabla 14. Caso de prueba Camino 3 (Fuente: Elaboración propia)..... - 47 -
Tabla 15. Caso de prueba Camino 4(Fuente: Elaboración propia)..... - 47 -
Tabla 16. Caso de prueba Camino 5. (Fuente: Elaboración propia)..... - 47 -
Tabla 17. Caso de prueba de Gestionar proyecto de curso. (Fuente: Elaboración propia) ..... - 48 -
Tabla 18. Resultado de la validación de la variable interpretabilidad. (Fuente: Elaboración propia) ..... - 56 -
Tabla 19. Cuadro lógico de Iadov. (Fuente: Elaboración propia)..... - 58 -
Tabla 20. Resultado de aplicación de la técnica Iadov. (Fuente: Elaboración propia)..... - 58 -
Tabla 21.Resultado de aplicación de la técnica Iadov. (Fuente: Elaboración propia)..... - 59 -

ÌNDICE DE IMÀGENES

Índice de imágenes

Figura 1.Secuencias de actividades del método propuesto. (Fuente: Peña Abreu y otros, 2018) ..... - 2 -
Figura 2. Conjunto Básico de Términos Lingüísticos. (Fuente: Peña Abreu y otros, 2018) ..... - 3 -
Figura 3. Modelo conceptual (Elaboración propia) ..... - 19 -
Figura 4. Modelo de datos. (Fuente: Elaboración propia) ..... - 29 -
Figura 5. Arquitectura de la herramienta mediante diagrama de componentes. (Fuente: Elaboración propia) ..... - 31 -
Figura 6. Patrón Experto en la entidad Estudiante. (Fuente: Elaboración propia) ..... - 31 -
Figura 7. Patrón Creador en la clase EstudianteController.php. (Fuente: Elaboración propia) ..... - 32 -
Figura 8. Patrón Controlador en la clase ProyectoController.php. (Fuente: Elaboración propia) ..... - 33 -
Figura 9. Uso del patrón Método de Fábrica. (Fuente: Elaboración propia) ..... - 33 -
Figura 10.Uso del patrón Adaptador en la clase Usuario.php. (Fuente: Elaboración propia) ..... - 34 -
Figura 11. Uso del patrón Decorador. (Fuente: Elaboración propia) ..... - 34 -
Figura 12. Representación de los atributos de calidad de la métrica TOC. (Fuente: Elaboración propia) ..... - 35 -
Figura 13. Representación de los atributos de calidad de la métrica RC. (Fuente: Elaboración propia) ..... - 35 -
Figura 14. Definición de clases. (Fuente: Elaboración propia) ..... - 44 -
Figura 15. Definición de variables. (Fuente: Elaboración propia) ..... - 44 -
Figura 16. Código del método TranformarDosTuplas (). (Fuente: Elaboración poropia) ..... - 45 -
Figura 17. Grafo de flujo. (Fuente: Elaboración propia) ..... - 45 -
Figura 18. Claustro de profesores. (Fuente: Elaboración propia) ..... - 53 -
Figura 19.Proyecto de curso I y estudiantes que lo conforman. (Fuente: Elaboración propia) ..... - 53 -
Figura 20. Ejemplo de algunos criterios considerados en la evaluación y pesos de los mismos. (Fuente: Elaboración propia) ..... - 54 -
Figura 21.Evaluación de estudiantes del proyecto "Proyecto I". (Fuente: Elaboración propia) ..... - 55 -
Figura 22. Evaluaciones de todos los profesores según los criterios. (Fuente: Elaboración propia) ..... - 55 -

INTRODUCCIÓN

En el contexto pedagógico se entiende por evaluación “aquel conjunto de procesos sistémicos de recogida, análisis e interpretación de información válida y fiable, que en comparación con una referencia o criterio nos permita llegar a una decisión que favorezca la mejora del objeto evaluado” (García, 1986).

Según plantea (MES, 2007) “La evaluación del aprendizaje es un proceso fundamental en el desarrollo del proceso docente educativo ya que comprueba el grado de cumplimiento de los objetivos que el estudiante debe vencer. Esta se estructura de forma frecuente, parcial, final y de culminación de los estudios. Cada una de estas tiene su forma particular de evaluarse. Pueden evaluarse mediante la observación del trabajo de los estudiantes, las preguntas orales y escritas, las pruebas parciales, los trabajos extraclases, los exámenes finales, la defensa de trabajos de diploma, los trabajos de curso, entre otros”.

En la Educación Superior uno de los tipos fundamentales de evaluación es el trabajo de curso. Estos son una forma de evaluación que incluye actividades que exigen una mayor complejidad que las de la evaluación frecuente ya que se miden criterios más globales dentro de los que se destacan: el grado de independencia logrado, profundidad alcanzada, completitud de las conclusiones, actualidad de referencias bibliográficas, proyección integral del estudiante, entre otros. (MES, 2007)

En esta forma de evaluación suelen conformarse tribunales. Según el artículo 179 de la Resolución No. 210/07 del Ministerio de Educación Superior “Estos tribunales están conformados por el jefe del departamento responsabilizado con esa actividad evaluativa y puede estar formado por profesores a tiempo completo, profesores a tiempo parcial y por especialistas de las entidades laborales del territorio, según sea necesario en cada uno de los casos”.

En el caso de las universidades cubanas, los tribunales están compuestos por varios profesores con diferentes niveles de experticia y diferentes categorías docentes. Estos pueden variar según sus responsabilidades y las asignaturas que imparten, lo cual indica la falta de homogeneidad en el proceso de evaluación.

El tribunal, para otorgar la calificación, puede considerar criterios tales como: la calidad del trabajo, la calidad de la exposición y la defensa por parte del estudiante, las opiniones del tutor y de la entidad laboral o de servicio donde se desarrolló el trabajo, entre otros aspectos (MES, 2007). Actualmente la medida más utilizada para evaluar el cumplimento de los objetivos de un estudiante han sido las notas de (5), (4), (3) y (2). Esta evaluación cuantitativa es determinista pero no siempre representa la información cualitativa de manera precisa, pues es muy difícil englobar todas las características del evaluado en este solo valor.

Por otra parte, en la actualidad debido al desarrollo de la tecnología se están utilizando medios para viabilizar los procesos de evaluación. La introducción de los avances científicos en la educación es imprescindible para alcanzar un desarrollo superior en la educación cubana. El desarrollo de las tecnologías de la información y las comunicaciones (TIC[1]) y sus redes de comunicación proporcionan nuevos métodos de enseñanza de las que el profesor universitario debe adueñarse. Estas han tenido un gran impacto en la organización del proceso de enseñanza y aprendizaje, la evaluación como elemento fundamental dentro este proceso, no escapa a esta realidad en la que las TIC constituyen

[1] **TIC:** Acrónimo en inglés de Tecnologías de la Información y las Comunicaciones.

un elemento que marca gran diferencia respecto a las prácticas de evaluación convencionales aplicadas a la enseñanza en ambientes de aprendizaje.

Según (Peña Abreu, y otros, 2018) en la universidad cubana actualmente el proceso de evaluación debería realizarse de forma diferente a épocas pasadas ya que las particularidades del estudiantado han cambiado, teniendo en cuenta las características de los mismos, marcados por ser nacidos en el siglo XXI, la era digital y del internet. Hoy el profesional que se necesita formar debe estar en consonancia con el momento histórico en que vive el país y el proceso docente educativo debe estar organizado y orientado al fin social que se pretende obtener.

La Universidad de las Ciencias Informáticas (UCI[2]) surge como una universidad de nuevo tipo. Una de las disciplinas para la formación del ingeniero en ciencias informáticas en dicha universidad es Ingeniería y Gestión de Software. En la misma existen varias asignaturas donde un componente fundamental en la evaluación del estudiante es el resultado del trabajo de curso. En la actualidad este se evalúa con esta misma medida cuantitativa, lo cual trae consigo una serie de insuficiencias como son:

- **Heterogeneidad en el claustro de profesores:** en esta evaluación participan varios profesores de la disciplina expresando su criterio en un solo dominio de expresión[3], dependiendo en gran medida de su conocimiento en correspondencia con su categoría y experiencia docente, lo cual puede producir imprecisión a la hora de emitir la nota.

- **Utilización de un solo dominio:** no se tiene en cuenta la posibilidad de que profesores emitan notas en diferentes dominios de expresión, lo cual trae consigo una nota poco flexible a la hora de evaluar un estudiante, así como poca interpretabilidad de los resultados. Esta interpretabilidad es necesaria a la hora de evaluar estudiantes que tienen una misma nota y no poseen el mismo grado de cumplimiento de las habilidades que demanda la asignatura para promover. Lo cual ayuda al profesor para un mejor acercamiento con el estudiante, logrando redefinir el proceso de aprendizaje del mismo a partir de la evaluación dada.

- **Objetividad a la hora de evaluar**: se evalúan muchos criterios los cuales miden el cumplimiento de los objetivos de la asignatura evaluada, así como su integración con otras asignaturas, lo cual puede resultar engorroso a la hora de evaluar, teniendo pérdida de información y lentitud en el proceso.

Algunos profesores del departamento de Ingeniería y Gestión de Software han investigado acerca de esta problemática. Entre sus trabajos se destaca el artículo propuesto por (Peña Abreu, y otros, 2018) el cual propone un método para evaluar de manera integral el aprendizaje en los trabajos de curso de la disciplina Ingeniería y Gestión de Software. Este permite la evaluación por parte de múltiples profesores utilizando múltiples criterios.

La realización de dicho método es de gran ayuda para los profesores, facilitando el proceso de toma de decisiones en el momento de emitir la evaluación final de los estudiantes, sin embargo, resulta engorroso su uso. Esto se debe a que es un proceso donde se llevan a cabo métodos matemáticos que son complicados de hacer de forma manual y su realización puede llevar mucho tiempo, teniendo en cuenta de deben aplicarse para cada estudiante que se evalúe

[2] **UCI**: Acrónimo de Universidad de las Ciencias Informáticas.
[3] **Dominio de expresión**: Formas de emitir evaluación por los profesores. Estos pueden ser: Intervalar, Numérico y Lingüístico.

y para los proyectos en general, lo cual puede producir pérdida de información, así como lentitud en el proceso evaluativo.

Es por esto que se propone, para solucionar las limitaciones anteriores, informatizar dicho método el cual considera la evaluación como un proceso donde intervienen varios actores que podrán emitir diversos criterios con diferentes pesos en un entorno de incertidumbre. Los criterios serán evaluados por dichos profesores, con la posibilidad de emitir notas en diferentes dominios de expresión. De esta forma se evita la pérdida de información que pueda existir, se agiliza el proceso de evaluación y se logra un mejor manejo de la incertidumbre en el proceso, lo cual contribuirá en gran medida a mejorar la calidad del mismo.

Teniendo en cuenta lo antes mencionado, queda definido como problema a resolver: las insuficiencias de las herramientas informáticas para la evaluación de trabajos de curso de la asignatura Ingeniería y Gestión de Software están afectando la interpretabilidad de los resultados para emitir la evaluación del estudiante.

Para la solución del problema descrito anteriormente se define como **objetivo general**: Desarrollar una herramienta informática para la evaluación de trabajos de curso basada en técnicas de soft computing que contribuya a mejorar la interpretabilidad de los resultados para emitir la evaluación del estudiante.

Para darle cumplimiento al objetivo general propuesto se han definido los siguientes **objetivos específicos:**

1. Construir el marco teórico de la investigación relacionado con las herramientas para evaluar los trabajos de curso, así como modelos, metodologías, herramientas y lenguajes para el desarrollo de software.

2. Desarrollar los artefactos ingenieriles para la obtención de la herramienta informática.

3. Comprobar la validez de la herramienta a partir de la validación de las variables dependientes y el correcto funcionamiento del software.

Se define como **objeto estudio:** desarrollo de software educativo, delimitado por el **campo de acción:** desarrollo de una herramienta informática para la evaluación de trabajos de curso basada en técnicas de soft computing.

Se tiene como **idea a defender:** si se desarrolla una herramienta para la evaluación de trabajos de curso basada en técnicas de soft computing entonces se contribuirá a la interpretabilidad de los resultados para emitir la evaluación del estudiante.

Con el objetivo de resolver el problema y lograr el objetivo planteado, se hizo necesaria la utilización de los siguientes **métodos de investigación**:

Métodos teóricos

- Histórico-lógico: se empleó para analizar la trayectoria y evolución de los modelos y herramientas existentes para la evaluación de trabajos de curso.

- Analítico-sintético: permitió realizar el estudio teórico de la investigación, lo que hace posible la selección de los elementos más importantes en el proceso de desarrollo de la herramienta y la selección de la literatura seleccionada para el tema a estudiar.

- Inductivo-deductivo: se utilizó para el razonamiento de la información consultada y llegar así a la obtención de un grupo de conocimientos particulares y generales.

- Modelación: permitió, junto con la metodología de desarrollo seleccionada, modelar los componentes esenciales para el desarrollo de la herramienta.

Métodos empíricos

- Entrevista: se utilizó en el intercambio con el usuario para adquirir información sobre el negocio y los requisitos que se deben cumplir en el desarrollo del software. Además, en las entrevistas a profesores del Departamento de Ingeniería y Gestión de Software de la UCI.

- Experimentación: por su importancia decisiva en las pruebas para demostrar el funcionamiento del sistema.

Estructura de la tesis

Capítulo I: Fundamentación teórica

En este capítulo se realiza una revisión y análisis de los modelos, métodos, metodologías y herramientas existentes para evaluar trabajos de curso. Se plantea la técnica de soft computing a utilizar computación con palabras y dentro de esta el modelo lingüístico 2-tupla. Se enuncian los principales conceptos relacionados con el tema, así como un estudio de las metodologías, lenguajes, herramientas y patrones a usar como apoyo para dar solución al problema planteado. Por último, se describen las pruebas a realizar para la validación y verificación del sistema.

Capítulo II: Análisis y diseño

En este capítulo se propone la solución de la investigación. Se exponen los requisitos funcionales y no funcionales. Se generan los artefactos del análisis y el diseño. Se describen los patrones de diseño arquitectónico empleados. Se valida el diseño aplicando las métricas Tamaño operacional de clases y Relaciones entre clases.

Capítulo III: Implementación y prueba.

En este capítulo se evalúa el cumplimiento de los objetivos planteados a partir de la realización de pruebas caja blanca y caja negra al sistema, pruebas de aceptación por parte del cliente y la validación de la solución propuesta a través de la introducción de un caso de estudio a la herramienta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se abordan los principales conceptos utilizados en la investigación. Se hace un estudio de diferentes modelos, métodos, metodologías y herramientas existentes, para la evaluación del aprendizaje de los trabajos de curso. Se caracteriza el método para la evaluación del aprendizaje en los trabajos de curso para la realización de procesos de computación con palabras. Se describen las principales tecnologías, lenguajes de programación y herramientas definidas para el desarrollo de la solución, así como la metodología de desarrollo, los patrones a emplear, las métricas para la validación del diseño y las pruebas para garantizar la calidad del producto final.

1.2 Principales conceptos

Para comprender los conceptos de evaluación, aprendizaje y *soft-computing* se presentan las siguientes definiciones:

Evaluación: Interpretación de la medida a partir de un patrón, que lleva a expresar un juicio de valor, acerca del proceso y los resultados de la enseñanza, el aprendizaje y la formación. (MES, 2007)

Aprendizaje: El aprendizaje se concibe como una adquisición estrictamente individual que incremental almacén explícito y declarativo de recursos mentales, “un saber decir-repetir”, en la creencia de que, aunque en el momento presente el aprendiz no encuentre su sentido o aplicabilidad ya lo encontrará en el

futuro. (PÉREZ GÓMEZ, 2010)

Evaluación del aprendizaje: La evaluación del aprendizaje es un proceso consustancial al desarrollo del proceso docente educativo. Tiene como propósito comprobar el grado de cumplimiento de los objetivos formulados en los planes y programas de estudio de la educación superior, mediante la valoración de los conocimientos y habilidades que los estudiantes van adquiriendo y desarrollando; así como, por la conducta que manifiestan en el proceso docente educativo. (MES, 2007).

Interpretabilidad: La interpretabilidad refleja la facilidad con que el usuario puede entender los datos y utilizarlos y analizarlos correctamente.

1.3 Métodos para la evaluación.

Entre los métodos existentes para la evaluación del aprendizaje se encuentran: la autoevaluación, la coevaluación y la heteroevaluación. Seguidamente se exponen los elementos distintivos de cada uno de ellos.

La **autoevaluación** se produce cuando el sujeto evalúa sus propias actuaciones, es un tipo de evaluación que toda persona realiza a lo largo de su vida; en el caso que nos ocupa, es de suma importancia que el alumno realice de manera continua ejercicios de valoración de su aprendizaje, de manera que le sea posible identificar aspectos que debe mejorar. En la medida en que un alumno logre contrastar sus avances contra estándares de actuación establecidos, podrá identificar áreas de mejora con lo cual estará en condiciones de regular su aprendizaje hacia el logro de competencias útiles para su desarrollo social y profesional (Leyva Barajas, 2010).

La **coevaluación** es una evaluación mutua, conjunta de una actividad o trabajo determinado entre varios estudiantes. En este caso, lo recomendable es que después de una serie de actividades didácticas, los participantes tanto alumnos como el profesor evalúen ciertos aspectos que consideren importantes de tal actuación conjunta. Es muy importante en la conducción de estos procesos de coevaluación pedir a los alumnos que se centren en la valoración tanto de los aspectos positivos o que ellos consideren como los más destacados, como en aquellos que es necesario trabajar más para mejorar la calidad del trabajo desarrollado en conjunto (Leyva Barajas, 2010).

La **heteroevaluación** consiste en la evaluación que realiza una persona sobre el trabajo, actuación o rendimiento de otra persona. Es aquella que habitualmente hace el profesor de sus alumnos. Dado que es un proceso importante e imprescindible de control en los esquemas y modelos educativos vigentes, rico por los datos y posibilidades que ofrece, delicado por el impacto que tiene en las personas evaluadas, y complejo por las dificultades técnicas que supone la emisión de juicios de valor válidos y objetivos (Leyva Barajas, 2010).

De estos métodos de evaluación el que más se asemeja a lo que se quiere obtener es la heteroevaluación, el cual se tomará como punto de partida para la realización del método propuesto por (Peña Abreu, y otros, 2018).

1.4 Método para la evaluación del aprendizaje en los proyectos de curso, utilizando Computación con palabras.

A continuación, se presenta el método (Peña Abreu, y otros, 2018), detallándose las actividades que se realizan en cada una de las fases descritas en la figura 1.

***Figura 1**.Secuencias de actividades del método propuesto. (Fuente: Peña Abreu y otros, 2018)*

Asumiendo que existe un conjunto de n estudiantes $E = \{e_i \mid i \in (1, \dots, n)\}$ que participan en el desarrollo de m proyectos de curso $T = \{t_j \mid j \in (1, \dots, m)\}$ y que en la evaluación de esos proyectos intervienen q profesores $P = \{q_l \mid l \in (1, \dots, q)\}$ se diseñan las siguientes actividades:

1 Definición de los criterios que se evaluarán

Los criterios deben ser definidos a partir de las habilidades establecidas para la disciplina de Ingeniería y Gestión de Software en el Plan de Estudios D de la carrera Ingeniería en Ciencias Informáticas. Los criterios serán definidos en un vector de la forma:

$$C = \{c_k \mid k \in (1, \dots, p)\}$$

2 Determinación de los pesos de los criterios

El peso que cada criterio tendrá en la evaluación, podrá variar de un estudiante a otro. Si se considera el recorrido del estudiante en la asignatura y el nivel en que ha desarrollado las habilidades requeridas, resulta útil poder evaluar un mismo criterio con dos niveles de importancia diferentes para dos estudiantes diferentes.

3 Recopilación de las evaluaciones individuales emitidas por los profesores

Para que los profesores emitan las evaluaciones, deben definirse los dominios en los que esa información puede expresarse. Los profesores podrán emitir sus valoraciones a través de diferentes dominios de expresión. La utilización de uno u otro estará condicionada -entre otros factores- por: la naturaleza de los criterios a evaluar y su nivel de conocimiento sobre el tema. Se propone en este proyecto el uso de los dominios numérico (N), intervalar (I) y lingüístico (S), que se especifican a continuación.

- Valores numéricos: $x_j^{ki} = v_j^{ki} \in [1,5]$
- Valores intervalares: $x_j^{ki} = I([0,1]) = \left[a_j^{ki}, b_j^{ki}\right] \; con \; a_j^{ki}, b_j^{ki} \in [0,1] \; y \; a_j^{ki} \leq b_j^{ki}$
- Valores lingüísticos: $x_j^{ki} = s_j^{ki} \in S = \{S_0, \ldots, S_g\}$ donde g+1 representa la cardinalidad del Conjunto de Términos Lingüísticos (CTL) S. Cada término lingüístico $S_i \in S$ tiene asociada una función de pertenencia $\mu_{S_i}(y), y \in [0,1]$.

4 Unificación de las evaluaciones individuales en un mismo dominio lingüístico

Las evaluaciones individuales emitidas por los profesores utilizando los dominios propuestos anteriormente deben ser unificadas en un mismo dominio. A partir de los sugerido por (Managing non-homogeneous information in group decision making, 2005) se unificarán sobre el dominio lingüístico. Como Conjunto Básico de Términos Lingüísticos (CBTL) se propone a $S_T = \{NE, C, M, R, B, MB, E\}$ cuya semántica se muestra en la Figura 2.

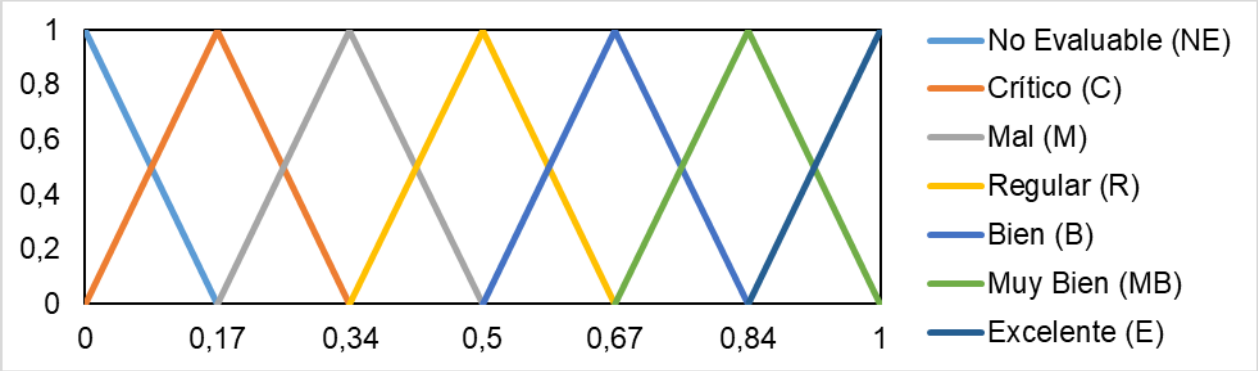


Figura 2.*Conjunto Básico de Términos Lingüísticos. (Fuente: Peña Abreu y otros, 2018)*

Luego de convertidas las evaluaciones de los profesores a conjuntos difusos, estos conjuntos deben ser transformados a 2-tuplas lingüísticas del CBTL definido anteriormente. Considerando las definiciones de traslación

simbólica y 2-tuplas dadas por (Herrera, et al., 2000), se utilizará como función de transformación la propuesta por (Martínez, y otros, 2012).

5 Agregación de la evaluación global de cada estudiante

5.1 Cálculo del valor colectivo de los criterios

Para calcular el valor colectivo de cada criterio para cada estudiante, considerando las evaluaciones emitidas por los profesores se utilizará el operador Media Aritmética Extendida (Herrera, et al., 2000), que significa el punto de equilibrio del conjunto de valores.

5.2 Agregar el valor de los criterios para cada estudiante

Para agregar el valor de los criterios de cada estudiante se utilizará el operador Media Ponderada Extendida (Herrera, et al., 2000), el cual permite agrupar los valores de los criterios considerando sus diferentes pesos.

6 Interpretación de los resultados

Una vez que se tienen los valores colectivos de los criterios y la evaluación final para cada uno de los estudiantes, se está en condiciones de analizarlos y tomar las decisiones apropiadas. Para realizar el análisis se utilizarán los operadores de comparación para 2-tuplas definidos en (Herrera, et al., 2000). Con estos operadores es posible analizar la información de diferentes maneras, lo que ofrece algunas facilidades para tomar la decisión final sobre la evaluación y para el proyecto docente-educativo. Algunas de esas ventajas son:

- Obtener la evaluación final integrada de cada estudiante: Cada criterio es ponderado y evaluado de manera particular para cada estudiante. Por lo que se obtiene una evaluación final del estudiante que es consistente.

- Determinar los estudiantes con mejores y peores resultados en cada una de las habilidades para orientar la atención diferenciada como parte de la entrega pedagógica.

- Determinar la evaluación global de cada proyecto de curso para orientar la investigación formativa del equipo, estimulando su presentación en eventos científicos estudiantiles y su publicación en revistas científicas, entre otras acciones.

1.5 Técnicas de Soft Computing

Hasta el momento de la definición de soft computing dada por Zadeh, la referencia a varias de las técnicas de la inteligencia artificial se hacía de manera independiente. Otros autores y el propio Zadeh (Zadeh, 1994) (Zadeh, 1998) han hecho reflexiones o definiciones sobre soft computing que de una manera u otra redundan en los mismos conceptos. Se puede considerar soft computing como un enfoque multidisciplinario donde la teoría de conjuntos borrosos facilita la representación del razonamiento humano y su capacidad de aprender en un ambiente de incertidumbre e imprecisión. La definición de soft computing está dada por medio de distintos conceptos y técnicas que buscan superar las dificultades y barreras que surgen en los problemas reales derivados de un mundo impreciso, incierto y difícil de categorizar. Una definición actual y acertada de soft computing, y en la cual se resume la esencia de lo que se pretende lograr con la investigación, es la dada por Verdegay," conjunto de técnicas y métodos que

permitan tratar las situaciones prácticas reales de la misma forma que suelen hacerlo los seres humanos, es decir, en base a inteligencia, sentido común, consideración de analogías, aproximaciones, etc…" (Verdegay, 2005).

1.5.1 Computación con palabras.

Un campo de investigación actual y una tendencia en el tratamiento de problemas de decisión es el uso de la computación con palabras. La CWW es uno de los paradigmas actuales para el tratamiento de la incertidumbre y la información lingüística. Es una técnica sobre la base de la lógica difusa donde las palabras se utilizan en lugar de números (Zadeh, 1996). Se presenta actualmente como uno de los paradigmas para el tratamiento de la incertidumbre y la información lingüística (Martínez, y otros, 2012) (Martínez, y otros, 2010). Tanto la CWW como las técnicas de soft computing tienen su base en los conceptos asociados a la teoría de conjuntos borrosos, la lógica borrosa y el modelado lingüístico de la información.

1.5.1.1 Modelos lingüísticos

Entre los modelos existentes para realizar procesos de CWW se encuentran:

- Modelos lingüísticos basados en funciones de pertenencia (MLCFP).

Es una representación lingüística de acuerdo a la definición de (Zadeh, 1998). Basado en funciones de pertenencia. Se definen las operaciones sobre las funciones de pertenencia, usando el principio de extensión. El resultado es un número borroso. Se realiza un proceso de aproximación para dar la respuesta en términos lingüísticos que produce pérdida de información.

- Modelos lingüísticos basados en conjuntos borrosos tipo-2 (MLCCBT2).

Este modelo computacional hace uso de conjuntos borrosos (difuso) tipo 2 para modelar evaluaciones lingüísticas. Es una representación basada en los conjuntos borrosos de tipo 2. Este reduce los esfuerzos computacionales para operar con conjuntos de tipo 2. El resultado se aproxima a un término lingüístico por lo que existe pérdida de información. Facilita la implementación de la CWW.

- Modelos lingüísticos basados en escalas ordinales (MLCSEO).

Representación lingüística de la información. Basado en el enfoque simbólico. Se utilizan

operadores de agregación basados en modelos simbólicos. Este modelo es muy utilizado en el paradigma de la CWW. El resultado se aproxima y por tanto hay pérdida de información.

- Modelos lingüísticos basados en 2-tuplas (MLC2T[4]).

Modela información lingüística basada en el concepto de traslación simbólica. Representación utilizando 2-tuplas. Se utilizan operadores de agregación simbólica y operadores numéricos redefinidos para operar con 2-tuplas. La

[4] **MLC2T**: Acrónimo de Modelos lingüísticos basados en 2-tuplas

representación empleando 2-tuplas permite mantener la desviación del valor real respecto al valor que se aproxima y, por tanto, evita la pérdida de información.

Tabla 1. *Comparación de los modelos lingüísticos. (Fuente: Elaboración propia)*

Modelos lingüísticos	**Pérdida de información**	**Evalúa en entornos de incertidumbre**	**Permite evaluación en diferentes dominios de expresión**
MLCFP	SI	SI	NO
MLCCBT2	SI	SI	NO
MLCSEO	SI	SI	NO
MLC2T	NO	SI	SI

Entre los modelos existentes se escogió modelo lingüístico 2- tuplas, ya que este evita la pérdida de información que se produce en los modelos anteriores de CWW. Permite una representación continua de la información lingüística en su dominio. También permite representar cualquier recuento de información obtenida en un proceso de agregación. Su modelo computacional basado en la traslación simbólica facilita los procesos de computación con palabras. Posee extensiones para diferentes contextos, entre ellos para la manipulación de información heterogénea.

1.6 Herramientas informáticas para la evaluación.

Se estudiaron además herramientas informáticas dedicadas a facilitar la toma de decisiones y los estudios de la evaluación del aprendizaje. A continuación, se realiza un análisis de algunas de ellas

EvalCOMIX:

Es un servicio web cuyo objetivo es que los estudiantes participen de forma activa en los procesos de evaluación de su aprendizaje. Es una herramienta gratuita, por lo que no tiene costo de licencia para su uso (García Vergara, 2014).

Quizalize:

Esta solución habilita la creación de exámenes on line desde cero, o tomando alguno creado por terceros, con las preguntas que los alumnos deben responder (en el aula o en casa). El docente puede seguir el proceso en tiempo real a través del Panel del Profesor, entregando puntos a los estudiantes que responden correctamente las preguntas en primer lugar. La puntuación total se entrega cuando todos los participantes hayan concluido. Es una herramienta privativa, por lo que tiene costo de licencia para su uso (TicherVirtual, 2016).

Net@nalisis:

Es una herramienta informática que muestra la representación gráfica del modelo de interpretación del estudiante o del grupo que desarrolla las actividades en el foro del EVA, dicha herramienta permite obtener indicadores individuales y

grupales según las medidas del análisis de redes sociales. Es una herramienta gratuita, por lo que no tiene costo de licencia para su uso. Solo es para evaluaciones sistemáticas (Se puede utilizar para analizar los modelos de interacción del estudiante y del grupo). No permite la evaluación de múltiples actores solo del profesor que evalúa la actividad (Tió Torriente, y otros, 2011).

1.6.1 Valoración de las herramientas

Luego de realizado el estudio de las herramientas de evaluación de trabajos mencionadas, se muestra una tabla comparativa entre estas herramientas y los indicadores definidos.

***Tabla 2**.Comparación de las herramientas de evaluación de trabajos estudiadas. (Fuente: Elaboración propia)*

Herramienta	**EvalCOMIX**	**Quizalize**	**net@nalisis**
Privativo	**SI**	**NO**	**NO**
Fines educativos	**SI**	**SI**	**SI**
Múltiples actores	**NO**	**NO**	**NO**
Dominios de expresión	**NO**	**NO**	**NO**
Agregar consenso	**SI**	**SI**	**SI**
Entornos de incertidumbre	**NO**	**SI**	**NO**
Pérdida de información	**NO**	**SI**	**SI**

Las herramientas analizadas son asociadas a fines educativos, lo cual constituye una de sus ventajas. Como se puede evidenciar en la tabla comparativa, una herramienta es privativa mientras que el resto no. No permiten la evaluación de múltiples actores, ni la evaluación en diferentes dominios de expresión. Permiten agregar consenso y solo una de ellas permite evaluar en entornos de incertidumbre y no tiene pérdida de información. Estas herramientas no satisfacen completamente las necesidades que posee actualmente la UCI de evaluar el aprendizaje de trabajos de curso. Por lo anteriormente expuesto, se propone el desarrollo de una herramienta informática para evaluar el aprendizaje de trabajos de curso, haciendo uso de la computación con palabras.

1.7 Metodología de desarrollo

Dentro del desarrollo de software y con la necesidad de que los proyectos lleguen al éxito y obtener un producto de gran valor para los clientes, se hace necesario el empleo de una metodología. La metodología seleccionada debe ser aquella que mejor se ajuste a las características del equipo de desarrollo y las exigencias de los usuarios finales.

Las metodologías de desarrollo se pueden enmarcar en dos grandes grupos, las metodologías tradicionales y las metodologías ágiles. Las tradicionales enfatizan en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto. Son recomendadas para proyectos de grandes dimensiones y con grandes equipos de desarrollo. Las

metodologías ágiles dan importancia a la capacidad de respuesta a los cambios, se enfatiza en la satisfacción del cliente y promueven el proyecto en equipo (Figueroa, y otros, 2010)

Para poder llevar a cabo el desarrollo de la propuesta de solución en un corto período de tiempo, donde el cliente esté en constante participación y colaboración y poder realizar cambios en los requisitos si fuese necesario, se opta por una metodología ágil de desarrollo de software en lugar de una metodología tradicional.

Se realizó un estudio de las metodologías ágiles XP, SCRUM y DSDM, teniendo en cuenta la respuesta a los cambios, el proyecto con el cliente, los planes de entrega de las iteraciones, el proyecto en equipo y el enfoque.

La **Programación Extrema (eXtreme Programming, XP)** es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el proyecto en equipo, preocupándose por el aprendizaje de los desarrolladores, y proporcionando un buen clima de proyecto. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisito imprecisos y muy cambiantes, y donde existe un alto riego técnico. (Metodologías ágiles de gestión de proyectos , 2008).

La **metodología SCRUM** define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprints es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. (Metodologías ágiles de gestión de proyectos , 2008).

Dynamic Systems Development Method (DSDM) es la metodología ágil más veterana y la que más se aproxima a los métodos tradicionales, su implantación incluso permitiría alcanzar un nivel 2 de madurez según CMMI. DSDM corrige los costos, la calidad y el tiempo desde el principio, deberes, aspectos clave y no tendrá que ajustar el producto del proyecto para cumplir con la restricción de tiempo establecido (Metodologías ágiles de gestión de proyectos , 2008).

Se selecciona como metodología de desarrollo la Programación Extrema (XP) por ser una metodología flexible a los cambios en las entregas, diseñada para equipos de proyecto pequeños donde la programación es por parejas, lo que cumple con las características del equipo de desarrollo. El cliente será partícipe del proceso de desarrollo estando en constante participación. Por parte del cliente se solicita que las entregas por iteraciones sean de un corto período de tiempo, con lo cual cumple esta metodología. Lo más importante tanto para el equipo de desarrollo como para el cliente es el producto final. Se generan los artefactos mínimos para la comunicación con el cliente. Esta consta de 4 fases: planificación, diseño, desarrollo y pruebas.

1.8 Herramientas, tecnologías y lenguajes

Se seleccionó como marco de proyecto Symfony3.1 pues el equipo de proyecto tenía experiencia en el desarrollo de aplicaciones usando este framework PHP. Se seleccionó como Gestor de Base de Datos, MySQL y el lenguaje MySQL para la implementación de los operadores de agregación del modelo lingüístico. Como Entorno de Desarrollo Integrado se seleccionó Netbeans v8.0 ya que es el IDE que mejor integración tiene con el marco de proyecto Symfony 3.1, permitiendo la ejecución de comandos y auto- completamiento de código. A continuación, se describen estas herramientas, tecnologías y lenguajes seleccionados:

1.8.1 Lenguaje de modelado

El **Lenguaje Unificado de Modelado** (*Unified Modeling Language*, UML) El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de software, en el flujo de procesos en la fabricación. Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene. UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML. UML guarda una relación directa con el análisis y el diseño orientados a objetos. (Lucid, 2018).

Se emplea el lenguaje de modelado UML en su versión 2.0 para la confección del diagrama de componentes y del modelo de datos de la solución.

1.8.2 Herramienta de modelado

Visual Paradigm for UML Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. (Herramienta para modelado UML, 2016)

Se selecciona Visual Paradigm en su versión 8.0 para la creación de los diagramas necesarios para el desarrollo de la solución.

1.8.3 Lenguaje de programación

PHP Hypertext Preprocessor (PHP) es un lenguaje de código abierto muy popular, adecuado para desarrollo web y que puede ser incrustado en HTML. Es popular porque un gran número de páginas y portales web están creadas con PHP. Este se utiliza para generar páginas web dinámicas. Recordar que llamamos página estática a aquella cuyos contenidos permanecen siempre igual, mientras que llamamos páginas dinámicas a aquellas cuyo contenido no es el mismo siempre (PHP, 2015).

Se seleccionó PHP en su versión 7, el cual tiene las siguientes características:

- Fácil manejo de errores: Utiliza *Engine Exceptions,* las que permiten reemplazar errores fatales con excepciones para manejar fácilmente los problemas.

- Soporta sistemas Windows de 64-Bit: Entrega soporte consistente a archivos de 64-Bits, por lo que funciona sin problemas en sistemas Windows.

- Libera espacio: Uno de los objetivos fundamentales de PHP 7 es liberar espacio para permitir optimizaciones.

- Nuevo operador de comparación: El operador nave espacial (< = >) permite comparar dos expresiones. Retorna 0 si los dos valores son iguales, 1 si el de la izquierda es mayor y -1 si el de la derecha es el mayor. (IDA, 2016).

1.8.4 Entorno de Desarrollo Integrado

Un IDE [5] (*Integrated Development Environment*) es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. Se caracteriza por ser un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

NetBeans es un IDE hecho principalmente para Java, pero puede ser utilizado para cualquier lenguaje de programación. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Es un producto libre y gratuito sin restricciones de uso. Facilita el proyecto mediante auto completamiento de código, visor de clases, métodos y componentes (NetBeans, 2015).

Se selecciona el IDE NetBeans en su versión 8.0 por ser un producto libre, además de brindar facilidades para la implementación de la solución.

Marco de trabajo

Un marco de trabajo o framework es un conjunto de componentes físicos y lógicos estructurados de manera que permiten ser reutilizados en el diseño y desarrollo de nuevos sistemas de información. Permiten la utilización de modelos arquitectónicos como Cuatro Capas y Modelo Vista Controlador (*Model View Controller*, MVC) (Recaman, 2012).

Symfony3.1 es uno de los frameworks más utilizados, versátiles y potentes del mercado. Sus componentes como el gestor de plantillas Twig, su ORM/DBAL Doctrine o su firme acogida a los estándares le han hecho merecedor de una posición privilegiada entre las opciones de las grandes empresas. Está tras plataformas tan potentes como phpBB, drupal o blackfire.

Características:

[5] IDE: Acrónimo de Entorno de Desarrollo Integrado.

- Symfony 3.1 es más estándar: (uso de PSR-3 para el registro y la eliminación de la interfaz Symfony \ Component \ HttpKernel \ Log \ LoggerInterface, ...)

- Corrige algunos errores de arquitectura: (los asistentes de consola con estado se eliminan a favor de mejores alternativas)

- Está más desacoplado y es más reutilizable que nunca (el HttpKernel se dividirá en varios más pequeños: el generador de perfiles, por ejemplo, será independiente, las clases se moverán de los paquetes a los componentes, los componentes se extraen de los existentes, ...) (Symfony, 2014)

1.8.5 Sistema Gestor de Base de Datos (SGBD)

MySQL es la base de datos de código abierto más popular del mundo. Con su rendimiento, confiabilidad y facilidad de uso comprobados, MySQL se ha convertido en la principal opción de base de datos para aplicaciones basadas en la Web, utilizada por propiedades web de alto perfil como Facebook, Twitter, YouTube.

Características:

- Velocidad. MySQL es rápido.

- Facilidad de uso. Es un sistema de base de datos de alto rendimiento, pero relativamente simple y es mucho menos complejo de configurar y administrar que sistemas más grandes.

- Es gratuito.

- Capacidad de gestión de lenguajes de consulta. MySQL comprende SQL, el lenguaje elegido para todos los sistemas de bases de datos modernos.

- Distribución abierta. Puede obtener y modificar el código fuente de MySQL. (Pérez García, 2007)

1.8.6 Servidor web

Generalmente, un usuario o navegador, solicita una página web cada vez que se conecta a Internet. El servidor se encarga de responder a esta “petición” o “demanda”, y manda el contenido solicitado por el usuario. Así pues, existen diferentes tipos de servidores HTTP, de entre ellos, algunos de los más popularmente conocidos y utilizados son: el servidor Apache, el servidor Microsoft IIS, y el servidor NGinx.(Mateu, 2004).

Apache es el más utilizado en el mundo digital, es el líder en este campo, gracias a que posee el mayor número de instalaciones a nivel mundial. Se caracteriza por ser un proyecto de código abierto y gratuito, es una multiplataforma que se utiliza para los sistemas operativos más importantes, es demasiado robusto y sobresale porque es el que ofrece mayor seguridad y rendimiento. (FireOS, 2017).

Se selecciona como servidor web Apache en su versión 2.4.4 por ser multiplataforma, cuenta con abundante documentación y soporta varios lenguajes de programación, dentro de los cuales se encuentra el seleccionado para la solución.

1.9 Patrones para el desarrollo de software

1.9.1 Patrón arquitectónico

Los patrones arquitectónicos representan el nivel más alto dentro del sistema de patrones y expresan el esquema de la estructura fundamental de la organización para sistemas de software (Almeira, y otros, 2007).

El patrón de arquitectura MVC permite realizar la programación multicapa, separando en tres componentes distintos los datos de una aplicación, la interfaz del usuario y la lógica de control. Este patrón se ve usualmente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el sistema de gestión de base de datos y el controlador representa la lógica del negocio, que está formado por tres niveles:

- **Modelo**: representa la información con la que trabaja la aplicación, o sea, su lógica de negocio.
- **Vista**: convierte el modelo en una página web que facilita al usuario interactuar con ella.
- **Controlador**: es el encargado de procesar las interacciones del usuario y ejecuta los cambios adecuados en el modelo o en la vista.

1.9.2 Patrones de diseño

Los patrones de diseño son una estructura de clases que se presenta en forma repetida en distintos diseños orientados a objetos, la cual es empleada para resolver un problema determinado de manera flexible y adaptable en forma dinámica (Almeira, y otros, 2007). Para el desarrollo del sistema se utilizaron varios patrones de diseño. A continuación, se dan a conocer los patrones empleados.

1.9.2.1 Patrones generales de software para asignación de responsabilidades (GRASP)

Los patrones GRASP constituyen un apoyo para la enseñanza que ayuda a uno a entender el diseño de objetos esencial, y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Este enfoque para la compresión y utilización de los principios de diseño se basa en los patrones de asignación de responsabilidades (Larman, 1999). A continuación, se mencionan los patrones utilizados en el diseño de la solución.

- **Experto**: es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Da origen a diseños donde el objeto de software realiza las operaciones que normalmente se aplican a la cosa real que representa, por lo que ofrece una analogía con el mundo real.
- **Creador**: guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.

- **Bajo Acoplamiento**: es un principio que se debe tener siempre en cuenta durante las decisiones de diseño. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Este patrón estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecienten la oportunidad de una mayor productividad. No puede considerarse en forma independiente de otros patrones como Experto o Alta cohesión, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades.

- **Alta Cohesión**: es la meta principal que ha de tenerse en cuenta en cada momento en todas las decisiones de diseño. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Una clase de alta cohesión posee un número relativamente pequeño, con una importante funcionalidad relacionada y poco trabajo que hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande.

- **Controlador:** La mayor parte de los Sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica para el usuario operado por una persona. Otros medios de entrada son los mensajes externos o las señales procedentes de sensores como sucede en los sistemas de control de procesos. En todos los casos, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. La misma clase controlador debería utilizarse con todos los eventos sistémicos de un caso de uso, de modo que se pueda conservar la información referente al estado del caso.

Patrones GoF[6]

Los patrones GoF pertenecen al campo del Diseño Orientado a Objetos. Están conformados por 23 patrones que se clasifican según su propósito en creacionales, estructurales y de comportamiento. (Christiansson, 2008)

- **Creacionales**: tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

- **Estructurales**: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.

- **Comportamiento**: Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

1.10 Métricas

Una métrica es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo determinado. (Pressman, 2010).

[6] GoF: del inglés Gand of Four

1.10.1 Métricas para diseño

Estas métricas cuantifican los atributos del diseño de manera que permiten evaluar su calidad. Dentro de las métricas para el diseño se encuentran las especializadas en diseño orientado a objetos, las cuales miden características de clases. Permiten averiguar cuan bien están definidas las clases y el sistema (Pressman, 2010). Miden de forma cuantitativa la calidad de los atributos internos del producto como son la responsabilidad de las clases, la reutilización, la complejidad de implementación y mantenimiento.

Se realizó un estudio de varias métricas de diseño que evaluaran en su totalidad los parámetros antes mencionados, entre estas están:

- **Árbol de Profundidad de Herencia (APH):** Se plantea sobre el árbol de herencia y mide la distancia desde el nodo hasta la hoja más lejana. Busca medir el grado de herencia que está fuertemente ligada a la reutilización.

- **Número de descendientes (ND):** Mide la calidad de la clase según la cantidad de descendientes que esta tenga. Utiliza como base para la determinación de la calidad, el concepto de que, si bien los descendientes indican reutilización, una cantidad elevada de descendientes puede diluir la abstracción utilizada para la creación de la súper clase.

- **Tamaño Operacional de Clase** (TOC[7]):

El tamaño operacional de una clase está dado por el número de métodos asignados a una clase. Para ello mide los siguientes atributos de calidad:

• **Responsabilidad**: responsabilidad asignada a una clase. Un aumento del TOC implica un aumento de esta responsabilidad, teniendo así demasiada responsabilidad, lo cual reduce la posibilidad de reutilización de la clase, lo que hace complicada la implementación y la prueba.

• **Complejidad de implementación**: grado de dificultad que tiene implementar un diseño de clases determinado. Un aumento del TOC implica un aumento de la complejidad de implementación de una determinada clase.

• **Reutilización:** grado de reutilización presente en una clase o estructura de clase. Un aumento del TOC implica una disminución del grado de reutilización de una determinada clase.

Tabla 3. *Criterios de evaluación para la métrica TOC. (Fuente: Elaboración propia)*

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

[7] **TOC**: Acrónimo de Tamaño Operacional de Clases.

implementación	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

- **Relaciones entre clases (RC):**

Esta métrica está dada por el número de relaciones de uso de una clase. Para ello mide los siguientes atributos de calidad:

- **Acoplamiento**: grado de dependencia de una clase o estructura de clase, con otras. Un aumento del RC provoca aumento del acoplamiento de la clase.

- **Complejidad de mantenimiento**: grado de esfuerzo necesario para desarrollar un arreglo, rectificación de algún error o mejora de un diseño. Un aumento del RC provoca aumento de la complejidad del mantenimiento de la clase.

- **Reutilización**: grado de reutilización presente en una clase o estructura de clase. Un aumento del RC provoca disminución en el grado de reutilización de la clase.

- **Cantidad de pruebas**: grado de esfuerzo necesario para realizar pruebas de unidad al sistema diseñado. Un aumento del RC provoca aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

***Tabla 4**.Criterios de evaluación para la métrica RC. (Fuete: Elaboración propia)*

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

	Alta	>2*Promedio

Se escogieron las métricas orientadas a clases Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC) (Kidd, y otros, 1994), que permiten evaluar estos parámetros y adicionalmente el bajo acoplamiento y la cantidad de pruebas de unidad que deben realizarse. Para evaluar las métricas son necesarios los valores de los umbrales para los parámetros de calidad. Algunos especialistas plantean umbrales basándose en el promedio de operaciones por clases obtenidos, estos valores fueron los aplicados en el diseño de la herramienta, se reflejan en Tabla 2 y Tabla 3.

1.11 Evaluación de software

La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación. Unas de las vías más importantes para determinar el estado de la calidad de un producto de software es el proceso de pruebas. Estas están dirigidas a componentes del sistema en su totalidad, con el objetivo de medir el grado en que cumple con los requerimientos.

Para la evaluación del software se realizan pruebas de caja blanca y pruebas de caja negra.

1.11.1 Pruebas de caja blanca

Las pruebas de Caja Blanca también suelen ser llamadas estructurales o de cobertura lógica. En ellas se pretende investigar sobre la estructura interna del código, exceptuando detalles referidos a datos de entrada o salida, para probar la lógica del programa desde el punto de vista algorítmico. Realizan un seguimiento del código fuente según se va ejecutando los casos de prueba, determinándose de manera concreta las instrucciones y bloques, que han sido ejecutados por los casos de prueba. En las pruebas de Caja Blanca se desarrollan casos de prueba que produzcan la ejecución de cada posible ruta del programa o módulo, considerándose una ruta como una combinación específica de condiciones manejadas por un programa. No todos los errores de software se pueden descubrir verificando todas las rutas de un programa, hay errores que se descubren al integrar unidades del sistema y pueden existir errores que no tengan relación con el código específicamente. (Cuervo Álvarez, 2015)

Estas se basan en el diseño de Casos de Prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante las pruebas de Caja Blanca el ingeniero de software puede obtener Casos de Prueba que:

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Las pruebas de Caja Blanca son consideradas entre las más importantes que se aplican a los sistemas, con la que se obtienen como resultados la disminución en un gran porciento el número de errores existentes en el software y por ende una mayor calidad y confiabilidad en la codificación. (Cuervo Álvarez, 2015)

1.11.2 Pruebas de caja negra

Las pruebas de Caja Negra también suelen ser llamadas funcionales y basadas en especificaciones. En ellas se pretende examinar el programa en busca de que cuente con las funcionalidades que debe tener y como lleva a cabo las mismas, analizando siempre los resultados que devuelve y probando todas las entradas en sus valores válidos e inválidos. Al ejecutar las pruebas de Caja Negra se desarrollan casos de prueba reales para cada condición o combinación de condiciones y se analizan los resultados que arroja el sistema para cada uno de los casos. En esta estrategia se verifica el programa considerándolo una caja negra. Las pruebas no se hacen en base al código, sino a la interfaz. No importa que se cubran todas las rutas dentro del programa, lo importante es probar todas las entradas en sus valores válidos e inválidos y lograr que el sistema tenga una interfaz amigable. Lograr una buena cobertura con pruebas de caja negra es un objetivo deseable; pero no suficiente a todos los efectos. Un programa puede pasar con holgura millones de pruebas de especificación y sin embargo tener defectos internos que surgen en el momento más inoportuno. Las pruebas de caja negra nos convencen de que un programa realizar bien sus funcionalidades programadas, pero no de que haga otras cosas menos aceptables. (Cuervo Álvarez, 2015)

1.12 Conclusiones parciales

Al finalizar el presente capítulo se arribó a las siguientes conclusiones parciales:

- La computación con palabras es el paradigma más conveniente a utilizar para el desarrollo de la herramienta informática en cuestión, para darle solución al problema planteado.

- Existen herramientas las cuales son utilizadas para la evaluación de trabajos, pero ninguna satisface las necesidades pertinentes como el trabajo con incertidumbre y la actuación de múltiples actores, por esto es necesario la utilización de una nueva herramienta.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

Introducción

En este capítulo se realiza la descripción de la propuesta de solución para facilitar su comprensión. En la primera fase o fase de planificación se identifican los requisitos funcionales y no funcionales, así como las técnicas para la obtención de los mismos. También se muestran las historias de usuario de los requisitos y el plan de iteraciones. Luego en la fase de diseño se presentan las tarjetas clase-responsabilidad-colaborador (CRC), el modelo de datos y la arquitectura del sistema. También se muestran los patrones de diseño a utilizar tanto GRASP como GoF. Finalmente se procede a la validación del diseño mediante las métricas TOC y RC.

Descripción general de la propuesta de solución

Para dar cumplimiento a los objetivos previamente planteados, el sistema propuesto permitirá la gestión de trabajos de curso y la adición de criterios, con los cuales poder realizar la evaluación de los mismos. Para realizar dicha evaluación el profesor principal debe introducir al sistema los proyectos que se desean evaluar, estableciendo los estudiantes que participan en el mismo, así como el claustro de profesores que realizará la evaluación. Luego de creados los proyectos el profesor principal debe establecer cuáles serán los criterios a evaluar en cada proyecto y asignar pesos a los mismos. Cuando el proyecto está listo los profesores que intervienen pueden establecer su evaluación a cada estudiante que conforma dicho proyecto, basándose en los criterios definidos en el dominio de expresión que deseen. Cuando todos los profesores terminan de evaluar cada estudiante el sistema realizará la unificación de las evaluaciones individuales en un mismo dominio, el lingüístico, para posteriormente realizar la agregación global de cada estudiante. Se obtendrá como resultado una evaluación general en formato de 2-Tuplas de cada estudiante, así como la evaluación general del trabajo realizado.

2.1 Roles relacionados con el sistema

Con el objetivo de restringir el acceso a las opciones que presenta el sistema, se definen los siguientes roles:

Tabla 5.*Roles relacionados con el sistema. (Fuente: Elaboración propia)*

Actores	Descripción
Administrador	Posee acceso a todo el sistema. Puede efectuar cualquier funcionalidad en el sistema.
Profesor principal	Se encarga de gestionar en el sistema los trabajos de curso y de realizar la evaluación. También se encarga de evaluar los trabajos atendiendo a los criterios que sean seleccionados.
Profesor	Se encarga de evaluar los estudiantes asociados a un trabajo de curso y de forma indirecta a este.

Modelo conceptual

El modelo conceptual, conocido también como modelo de dominio es la descripción de cómo se relacionan los conceptos en un problema. El modelo conceptual sirve para representar un problema de manera gráfica a través de diagramas entidad relación, diccionarios/glosarios y diagrama de clases. Es importante para abstraer un problema e

identificar como interactúa el sistema en el cual se desenvuelve la solución. Al modelar un problema se identifica su funcionamiento y es realizado para solucionar problemas. (Juan Pablo, 2013)

La Figura 3 muestra los principales conceptos del dominio y las relaciones entre ellos.

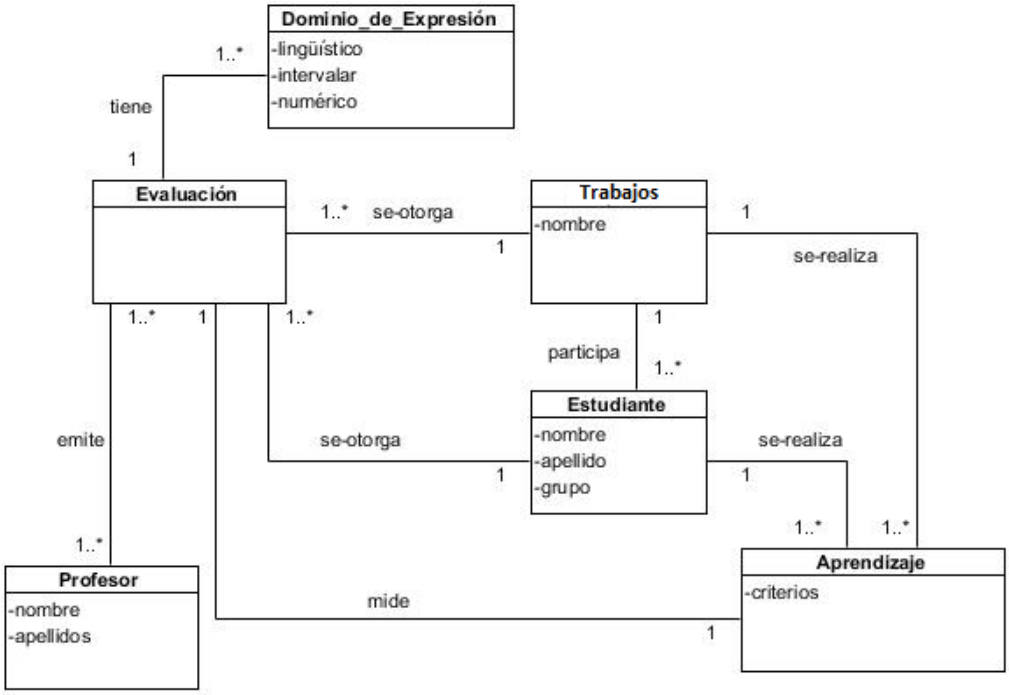


Figura 3. *Modelo conceptual (Elaboración propia)*

Descripción de los conceptos

Trabajos: Representa los trabajos a evaluar.

Estudiantes: Representa los estudiantes a evaluar.

Evaluación: Representa la evaluación de los trabajos y estudiantes haciendo uso del modelo lingüístico 2-tuplas.

Aprendizaje: Representa el método de evaluación del aprendizaje de los trabajos y estudiantes. Utilizando el método para la evaluación del aprendizaje en los trabajos de curso, aplicando computación con palabras. (Peña Abreu, y otros, 2018)

Profesor: Personal capacitado para evaluar.

Dominio de Expresión: Representa los dominios de expresión que tienen en cuenta los profesores para emitir la evaluación.

2.2 Fase de planificación

Para el desarrollo de la herramienta se dividirá el proyecto en las 4 fases propuestas por la metodología XP, planificación, diseño, desarrollo y pruebas.

En la fase de planificación se efectúa un diálogo entre las partes involucradas en el proyecto, incluyendo al cliente y a los programadores. Se comienza recopilando las historias de usuario, las cuales sustituyen a los casos de uso y se evalúa el tiempo de desarrollo de cada una por parte de los programadores. Mediante un plan de iteraciones se planifica el tiempo que dura el desarrollo del sistema.

2.2.1 Requisitos del software

Como parte del análisis previo al diseño de la solución se emplearon las siguientes técnicas de obtención o captura de requisitos:

La **entrevista** es de gran utilidad para obtener información cualitativa como opiniones, o descripciones subjetivas de actividades. Es una técnica muy utilizada, y requiere una mayor preparación y experiencia por parte del analista. La entrevista se puede definir como un "intento sistemático de recoger información de otra persona" a través de una comunicación interpersonal que se lleva a cabo por medio de una conversación estructurada. Debe quedar claro que no basta con hacer preguntas para obtener toda la información necesaria. Es muy importante la forma en que se plantea la conversación y la relación que se establece en la entrevista (Arturo Guerra, 2018).

La **tormenta de ideas** consiste en reuniones con cuatro a diez personas donde como primer paso sugieren toda clase de ideas sin juzgar su validez –por muy disparatadas que parezcan–, y después de recopilar todas las ideas se realiza un análisis detallado de cada propuesta. Esta técnica se puede utilizar para identificar un primer conjunto de requisitos en aquellos casos donde no están muy claras las necesidades que hay que cubrir, o cuando se está creando un sistema que habilitará un servicio nuevo para la organización. (Arturo Guerra, 2018)

2.2.1.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de servicios que el sistema debería proporcionar, cómo el sistema debería reaccionar a entradas particulares, y cómo debería comportarse el sistema en situaciones particulares. En algunos casos, los requisitos funcionales también pueden indicar explícitamente lo que el sistema no debería hacer. (Sommerville, 2011)

Luego de aplicadas las técnicas de entrevista y tormenta de ideas se identificaron un total de 69 requisitos funcionales (RF). La prioridad de cada requisito fue definida por el cliente en función de la importancia para el negocio.

Tabla 6. *Requisitos funcionales. (Fuente: Elaboración propia)*

Descripción	Prioridad
RF-1. Adicionar profesor	Media
RF-2. Modificar profesor	Media
RF-3. Eliminar profesor	Media
RF-4. Listar profesor	Media
RF-5. Buscar profesor	Media

RF-6. Mostrar profesor	Media
RF-7. Adicionar estudiante	Alta
RF-8. Modificar estudiante	Alta
RF-9. Eliminar estudiante	Alta
RF-10. Listar estudiante	Baja
RF-11. Buscar estudiante	Baja
RF-12. Adicionar Proyecto de curso	Alta
RF-13. Modificar Proyecto de curso	Alta
RF-14. Eliminar Proyecto de curso	Alta
RF-15. Listar Proyecto de curso	Baja
RF-16. Buscar Proyecto de curso	Baja
RF-17. Modificar escala de Saaty	Alta
RF-18. Ordenar Criterio	Alta
RF-19. Combinar Valoraciones	Alta
RF-20. Evaluar Proyectos	Alta
RF-21. Adicionar objetivos de la asignatura	Alta
RF-22. Modificar objetivos de la asignatura	Media
RF-23. Eliminar objetivos de la asignatura	Media
RF-24. Listar objetivos de la asignatura	Baja
RF-25. Buscar objetivos de la asignatura	Baja
RF-26. Unificar evaluación a un mismo dominio	Alta
RF-27. Transformar evaluación a 2-tuplas	Alta
RF-28. Agregar evaluación global de cada estudiante	Alta
RF-29. Calcular valor colectivo de cada criterio para cada proyecto	Alta
RF-30. Agregar valor de los criterios para cada estudiante	Alta
RF-31. Ordenar evaluación de cada estudiante	Media
RF-32. Adicionar rol	Media
RF-33. Modificar Rol	Media
RF-34. Eliminar Rol	Media

RF-35. Listar Roles	Baja
RF-36. Buscar Rol	Baja
RF-37. Adicionar Evaluación a estudiante	Alta
RF-38. Eliminar evaluación de estudiante	Media
RF-39. Modificar Evaluación de estudiante	Media
RF-40. Mostrar resultados de la evaluación	Media
RF-41. Listar evaluaciones de estudiantes	Baja
RF-42. Buscar evaluaciones de estudiantes	Baja
RF-43. Generar listado ordenado de proyectos evaluados	Alta
RF-44. Mostrar resultado de la evaluación	Alta
RF-45. Adicionar Nomenclador Asignatura	Media
RF-46. Adicionar Nomenclador Categoría docente	Media
RF-47. Adicionar Nomenclador Grado científico	Media
RF-48. Modificar Nomenclador Asignatura	Media
RF-49. Modificar Nomenclador Categoría docente	Media
RF-50. Modificar Nomenclador Grado científico	Media
RF-51. Eliminar Nomenclador Asignatura	Baja
RF-52. Eliminar Nomenclador categoría docente	Baja
RF-53. Eliminar Nomenclador Grado científico	Baja
RF-54. Listar Nomenclador Asignatura	Baja
RF-55. Listar Nomenclador Categoría docente	Baja
RF-56. Listar Nomenclador Grado científico	Baja
RF-57. Mostrar Nomenclador Asignatura	Media
RF-58. Mostrar Nomenclador Categoría docente	Media
RF-59. Mostrar Nomenclador Grado científico	Media
RF-60. Buscar Nomenclador Asignatura	Baja
RF-61. Buscar Nomenclador Categoría docente	Baja
RF-62. Buscar Nomenclador Grado científico	Baja
RF-63. Adicionar Profesor a un proyecto	Alta

RF-64.	Eliminar Profesor de un proyecto	Baja
RF-65.	Adicionar Estudiante a un proyecto	Alta
RF-66.	Eliminar Estudiante de un proyecto	Baja
RF-67.	Adicionar Criterio a un proyecto	Alta
RF-68.	Eliminar Criterio de un proyecto	Baja
RF-69.	Mostrar dominio de expresión	Media

2.2.1.2 Requisitos no funcionales

Los requisitos no funcionales son restricciones en los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, restricciones en el proceso de desarrollo. Los requisitos no funcionales a menudo se aplican al sistema como un todo, en lugar de un sistema individua.. (Sommerville, 2011)

Tabla 7. *Requisitos no funcionales. (Fuente: Elaboración propia)*

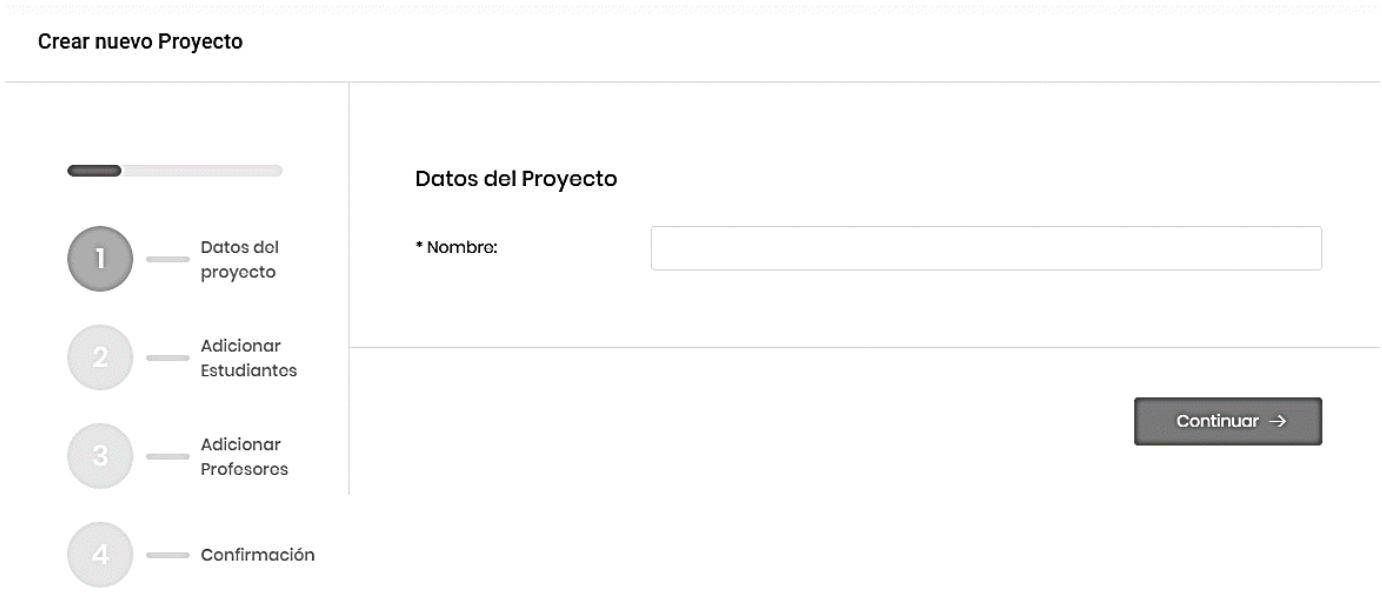
Código	**Descripción**
Usabilidad	
RNF 1	La herramienta informática a desarrollar será una aplicación web.
RNF 2	La aplicación debe presentar una vista sencilla, descriptiva y fácil de usar, con el objetivo de proporcionar a los usuarios un mejor entendimiento con la aplicación.
Portabilidad	
RNF 3	La aplicación debe poder instalarse en los sistemas operativos Windows y Linux/Unix
Fiabilidad	
RNF 4	La aplicación restringirá el acceso por roles a los usuarios autorizados.
Disponibilidad	
RNF 5	La aplicación deberá estar disponible siempre, asegurando el acceso desde cualquier lugar de red a los usuarios autorizados.
Eficiencia	
RNF 6	El tiempo de respuesta en el procesamiento de los datos y solicitud de estos no debe sobrepasar los 5 segundos.
Interfaz	
RNF 7	La aplicación debe ofrecer una interfaz amigable y un diseño sencillo que le permita al usuario interactuar fácilmente.

RNF 8	La aplicación, al ocurrir un error con los datos de entrada muestra mensajes al usuario informando este.
Software	
RNF 9	El servidor de aplicación debe tener instalado el servidor web Apache y el marco de proyecto Symfony.
RNF 10	El servidor de base de datos debe tener como Gestor de Base de Datos MySQL.
RNF 11	Las PC clientes deben tener instalado un navegador web (Mozilla Firefox, Google Chrome); se recomienda para estas Mozilla Firefox en su versión 24.0 o superior.
Hardware	
RNF 12	La PC servidor debe tener 2 GB de memoria RAM o superior, un disco duro de 320 GB de almacenamiento o superior, un procesador Pentium IV 2.4 GHz o superior.
RNF 13	Las PC clientes deben tener 512 MB, como mínimo de memoria RAM, un procesador Pentium IV 1.7 GHz o superior.
Seguridad	
RNF 14	Existirán diferentes tipos de usuarios según las acciones que puedan realizar. Los permisos serán limitados, basados en los roles definidos y el usuario no podrá gestionarlos.
RNF 15	La aplicación debe permitir que la contraseña se almacene de manera encriptada en la base de datos.
RNF 16	La aplicación debe permitir la comprobación de credenciales en la autenticación del usuario en el servidor de aplicaciones y no en el servidor de base de datos, para evitar inyecciones SQL que falseen la autenticación y provoquen la suplantación de identidad.
Implementación	
RNF 17	Los operadores de agregación del modelo 2-tuplas deberán ser implementados en el lenguaje php.
RNF 18	La herramienta informática será desarrollada con HTML5 y haciendo uso de CSS y JavaScript.

2.2.2 Historias de usuario

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Las mismas son escritas por los clientes como las tareas que el sistema debe hacer y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas.

A continuación, se muestra la historia de usuario: Adicionar proyecto de curso, el resto de las historias de usuario se pueden observar en los Anexos.

<table>
<tr><td colspan="2">HISTORIA DE USUARIO</td></tr>
<tr><td>Número: HU-12</td><td>Nombre historia de usuario: Adicionar proyecto de curso</td></tr>
<tr><td>Usuario: Jenniffer Maulini</td><td>Iteración asignada: 1</td></tr>
<tr><td>Prioridad en negocio: Alta</td><td>Puntos estimados: 3 días</td></tr>
<tr><td>Riesgo en desarrollo:</td><td>Puntos reales 3 días</td></tr>
<tr><td colspan="2">Descripción:
Permite al especialista adicionar un proyecto de curso en el sistema. Se debe mostrar una interfaz para el proyecto de curso.</td></tr>
<tr><td colspan="2">Observaciones:
Esta funcionalidad solo puede ser realizada por los roles especialista y administrador.</td></tr>
<tr><td colspan="2">Interfaz de usuario:
 
</td></tr>
</table>

2.2.3 Plan de iteraciones

Las historias de usuarios seleccionadas para cada plan de entrega son desarrolladas y probadas en un ciclo de iteración de acuerdo al orden preestablecido donde se estima el tiempo de duración de cada una de ellas. Cada historia de usuario se traduce en tareas específicas de programación. (Joskowicz, 2012). El sistema será desarrollado

en tres iteraciones, en una primera iteración serán implementadas las historias de usuarios de prioridad alta para el negocio, en una segunda iteración las de prioridad media y en una tercera iteración las de prioridad baja.

Tabla 8. *Plan de iteraciones. (Fuente: Elaboración propia)*

Iteración	Historia de usuario	Prioridad	Duración de HU (días)	Duración total (semanas)
1	Adicionar estudiante	Alta	3	13
	Modificar estudiante	Alta	4	
	Eliminar estudiante	Alta	3	
	Adicionar proyecto de curso	Alta	3	
	Modificar proyecto de curso	Alta	4	
	Eliminar proyecto de curso	Alta	3	
	Modificar escala de Saaty	Alta	5	
	Ordenar criterio	Alta	3	
	Combinar valoraciones	Alta	5	
	Evaluar proyectos	Alta	4	
	Adicionar objetivos de la asignatura	Alta	4	
	Unificar evaluación a un mismo dominio	Alta	5	
	Transformar evaluación a 2-Tuplas	Alta	5	
	Agregar evaluación global de cada proyecto	Alta	5	
	Calcular valor colectivo de cada criterio para cada estudiante	Alta	5	
	Agregar valor de los criterios para cada estudiante	Alta	4	
	Adicionar evaluación de estudiante	Alta	4	
	Generar ordenado de proyectos evaluados	Alta	5	
	Mostrar resultados de la evaluación	Alta	6	
	Adicionar profesor a un proyecto	Alta	3	
	Adicionar estudiante a un proyecto	Alta	3	
	Adicionar criterio a un proyecto	Alta	3	
	Adicionar profesor	Media	2	

2	Modificar profesor	Media	2	7
	Eliminar profesor	Media	1	
	Listar profesor	Media	1	
	Buscar profesor	Media	2	
	Mostrar profesor	Media	2	
	Modificar objetivos de la asignatura	Media	3	
	Eliminar objetivos de la asignatura	Media	2	
	Ordenar evaluación de cada estudiante	Media	3	
	Adicionar rol	Media	2	
	Modificar rol	Media	3	
	Eliminar rol	Media	2	
	Eliminar evaluación de estudiante	Media	2	
	Modificar evaluación de estudiante	Media	3	
	Mostrar resultados de la evaluación	Media	3	
	Adicionar Nomenclador Asignatura	Media	2	
	Adicionar Nomenclador Categoría docente	Media	2	
	Adicionar Nomenclador Grado científico	Media	2	
	Modificar Nomenclador Asignatura	Media	2	
	Modificar Nomenclador Categoría docente	Media	2	
	Modificar Nomenclador Grado científico	Media	2	
	Mostrar Nomenclador Asignatura	Media	1	
	Mostrar Nomenclador Categoría docente	Media	1	
	Mostrar Nomenclador Grado científico	Media	1	
	Mostrar dominio de expresión	Media	1	
	Listar estudiante	Baja	1	
	Buscar estudiante	Baja	1	
	Listar proyecto de curso	Baja	1	
	Buscar proyecto de curso	Baja	1	
	Listar objetivos de la asignatura	Baja	2	

3	Buscar objetivos de la asignatura	Baja	2	5
	Listar rol	Baja	1	
	Buscar rol	Baja	1	
	Listar evaluaciones de estudiantes	Baja	1	
	Buscar evaluaciones de estudiantes	Baja	2	
	Eliminar Nomenclador Asignatura	Baja	2	
	Eliminar Nomenclador Categoría docente	Baja	2	
	Eliminar Nomenclador Grado científico	Baja	2	
	Listar Nomenclador Asignatura	Baja	1	
	Listar Nomenclador Categoría docente	Baja	1	
	Listar Nomenclador Grado científico	Baja	1	
	Buscar Nomenclador Asignatura	Baja	2	
	Buscar Nomenclador Categoría docente	Baja	2	
	Buscar Nomenclador Grado científico	Baja	2	
	Eliminar profesor de un proyecto	Baja	2	
	Eliminar estudiante de un proyecto	Baja	2	
	Eliminar criterio de un proyecto	Baja	2	
Total				**25**

2.3 Fase de diseño

En la fase de diseño se confeccionan las tarjetas Clase Responsabilidad Colaborador (CRC) para crear diseños de clases orientados a responsabilidades. Se selecciona la arquitectura adecuada para el desarrollo de la herramienta.

2.3.1 Tarjetas clase-responsabilidad-colaborador (CRC)

La utilización de tarjetas CRC (Class-Responsibility-Collaboration) es una técnica de diseño orientado a objetos. Cuyo objetivo es hacer, mediante tarjetas, un inventario de las clases que vamos a necesitar para implementar el sistema y la forma en que van a interactuar. De esta forma se pretende facilitar el análisis y discusión de las mismas por parte de varios actores del equipo de proyecto con el objetico de que el diseño sea lo más simple posible verificando las especificaciones del sistema. (Jummp, 2012).

A continuación, se muestra la tarjeta CRC de la clase: Proyecto, el resto de las tarjetas se pueden observar en los Anexos.

Tabla 9. *Tarjeta CRC de la clase Proyecto. (Fuente: Elaboración propia)*

TARJETA CRC	
Clase: Proyecto	
Descripción: Almacena los Proyectos de curso.	
Responsabilidades	**Colaboradores**
Contener la información de los proyectos de curso.	1. Ncategoria 2. NgradoCientifico 3. Estudiante 4. Criterio 5. Dominio 6. Profesor

2.3.2 Modelo de datos

El modelado de datos es el proceso de documentar un diseño de sistema de software complejo como un diagrama de fácil comprensión, usando texto y símbolos para representar la forma en que los datos necesitan fluir. El diagrama se puede utilizar como un mapa para la construcción de un nuevo software o para la reingeniería de una aplicación antigua. Tradicionalmente, los modelos de datos se han construido durante las fases de análisis y diseño de un proyecto, para asegurar que los requisitos para una nueva aplicación se entienden completamente. Un modelo de datos puede ser pensado como un diagrama de flujo que ilustra las relaciones entre los datos. A pesar de que la captura de todas las posibles relaciones en un modelo de datos puede consumir mucho tiempo, es un paso importante que no debería ser apresurado. Los modelos de datos físicos, lógicos y conceptuales bien documentados permiten que las partes interesadas identifiquen errores y hagan cambios antes de que cualquier código de programación se haya escrito. Los modeladores de datos suelen utilizar varios modelos para ver los mismos datos y garantizar que todos los procesos, entidades, relaciones y flujos de datos han sido identificados. (TechTarget, 2016).

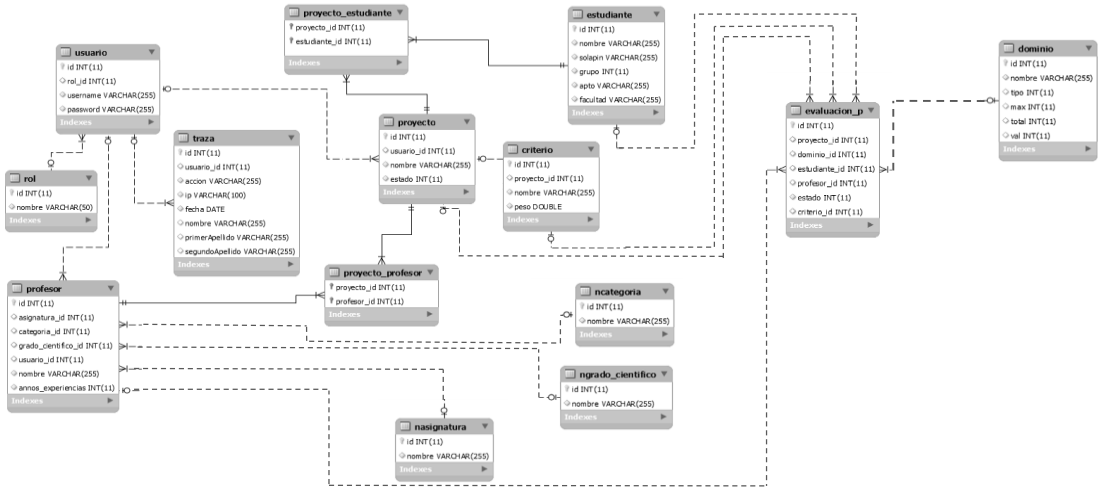


Figura 4. *Modelo de datos. (Fuente: Elaboración propia)*

2.3.3 Arquitectura del sistema

La arquitectura de la herramienta está basada en el patrón arquitectónico Modelo Vista Controlador, en el cual está basado el marco de proyecto seleccionado Symfony2. Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

- El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

- La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.

- El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Para mostrar la estructura y composición de la herramienta haciendo uso de este patrón arquitectónico se realiza un diagrama de componentes, donde se muestra la relación entre estos.

Lo que distingue a un diagrama de componentes de otros tipos de diagramas es su contenido. Normalmente contienen componentes, interfaces y relaciones entre ellos. Y como todos los diagramas, también puede contener paquetes utilizados para agrupar elementos del modelo. Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. Dado que los diagramas de componentes muestran los componentes software que constituyen una parte reusable, sus interfaces, y sus interrelaciones, en muchos aspectos se puede considerar que un diagrama de componentes es un diagrama de clases a gran escala. Cada componente en el diagrama debe ser documentado con un diagrama de componentes más detallado, un diagrama de clases, o un diagrama de casos de uso. Un paquete en un diagrama de componentes representa una división física del sistema. Los paquetes se organizan en una jerarquía de capas donde cada capa tiene una interfaz bien definida. Un ejemplo típico de una jerarquía en capas de este tipo es: Interfaz de usuario; Paquetes específicos de la aplicación; Paquetes reusables; Mecanismos claves; y Paquetes hardware y del sistema operativo. Un diagrama de componentes se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (generalmente de compilación). Puede mostrar también que un componente software contiene una interfaz, es decir, la soporta. (UNAD, 2013).

La Figura 5 muestra la arquitectura de la herramienta a través del diagrama de componentes. Este está compuesto por un componente principal llamado GestionBundle, el cual está asociado a otros componentes. Se puede apreciar el componente Modelo, donde se encuentran las clases Repositorios, que hacen uso del componente ORM Doctrine,

y las clases Entidades. La Vista hace uso de los componentes Twig. Esta a su vez es usada por el Controlador, el cual utiliza los componentes Enrutamiento y Entidades.

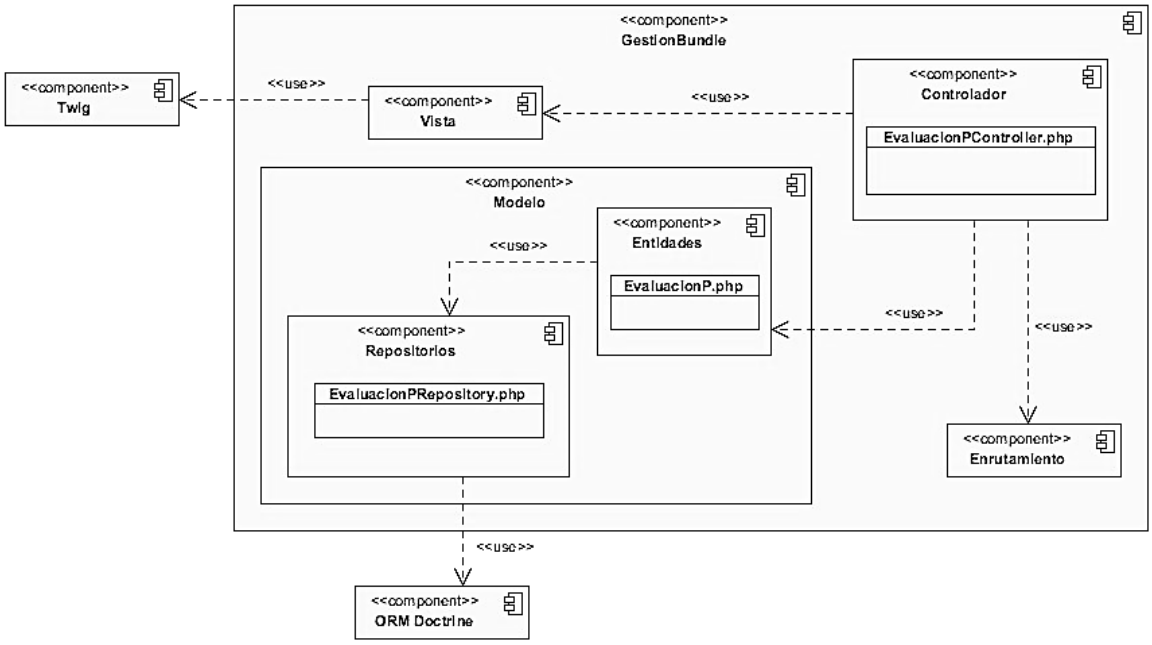


Figura 5. *Arquitectura de la herramienta mediante diagrama de componentes. (Fuente: Elaboración propia)*

2.4 Patrones de diseño

2.4.1 Patrones generales de software para asignación de responsabilidades (GRASP)

A continuación, se describen los patrones GRASP empleados en el desarrollo del sistema.

- **Experto**: el uso de este patrón resulta de utilidad en las clases del modelo, las cuales contienen toda la información relacionada con los objetos persistentes que representan. La Ilustración 4 muestra el uso de este patrón en la entidad Estudiante.

```
private $nombre;

/**
 * @return string
 */
public function getNombre()
{
    return $this->nombre;
}

/**
 * @param string $nombre
 */
public function setNombre($nombre)
{
    $this->nombre = $nombre;
}
```

Figura 6.*Patrón Experto en la entidad Estudiante. (Fuente: Elaboración propia)*

- **Creador**: este patrón se pone de manifiesto en las clases controladoras del sistema. Brinda soporte a un bajo acoplamiento y mejora la reutilización. La Ilustración 5 muestra el uso de este patrón en la clase EstudianteController.php.

```
public function newAction(Request $request)
{
    $estudiante = new Estudiante();
    $form = $this->createForm('GestionBundle\Form\EstudianteType', $estudiante);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $em->persist($estudiante);
        $em->flush();

        return $this->redirectToRoute('estudiante_show', array('id' => $estudiante->getId()));
    }

    return $this->render('estudiante/new.html.twig', array(
        'estudiante' => $estudiante,
        'form' => $form->createView(),
    ));
}
```

Figura 7. *Patrón Creador en la clase EstudianteController.php. (Fuente: Elaboración propia)*

- **Bajo acoplamiento**: en el diseño del sistema cada clase depende lo menos posible de otra, de tal manera que una de ellas solo recurre a otra en caso de que exista referencia dentro de sus atributos, lo que permite que el sistema sea mucho más robusto y de fácil mantenimiento. Este se evidencia en el marco de proyecto, ya que dentro de la capa modelo las clases de abstracción de datos son las más reutilizables y no tienen asociaciones con las clases de la capa vista ni con el controlador.

- **Alta cohesión:** en el diseño del sistema las clases tienen responsabilidades estrechamente relacionadas y no realizan un proyecto excesivo, lo que permite simplificar el mantenimiento y aumentar la capacidad de reutilización de estas. Symfony permite asignar responsabilidades con una alta cohesión. Este se evidencia en las clases repositorio, que tienen como única responsabilidad realizar operaciones de acceso a datos solo con la entidad que representa.

- **Controlador**: este patrón se evidencia en las clases controladoras, responsables de implementar las funcionalidades pertenecientes a una interfaz determinada. La Ilustración 6 muestra el uso de este patrón en la clase controladora ProyectoController.php.

```
class ProyectoController extends Controller
{
    /** Lists all Proyecto entities. ...*/
    public function indexAction(){...}

    /** Creates a new Proyecto entity. ...*/
    public function newAction(Request $request){...}

    /** Finds and displays a Proyecto entity. ...*/
    public function showAction(Proyecto $proyecto){...}

    /** Displays a form to edit an existing Proyecto entity. ...*/
    public function editAction(Request $request, Proyecto $proyecto){...}

    /** Deletes a Proyecto entity. ...*/
    public function deleteAction(Request $request, Proyecto $proyecto){...}

    /** Creates a form to delete a Proyecto entity. ...*/
    private function createDeleteForm(Proyecto $proyecto){...}
}
```

Figura 8.*Patrón Controlador en la clase ProyectoController.php. (Fuente: Elaboración propia)*

2.4.2 Patrones GoF aplicados

A continuación, se describen los patrones GoF utilizados en el desarrollo del sistema.

- **Creacionales**:

Método de Fábrica (del inglés Factory Method): define una interfaz para la creación de un objeto, lo que permite a las subclases decidir de qué clase instanciarlo. Permite que una clase difiera la instanciación en favor de sus subclases.

```
public function newAction(Request $request)
{
    $proyecto = new Proyecto();
    $form = $this->createForm('GestionBundle\Form\ProyectoType', $proyecto);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $em->persist($proyecto);
        $em->flush();

        return $this->redirectToRoute('proyecto_show', array('id' => $proyecto->getId()));
    }

    return $this->render('proyecto/new.html.twig', array(
        'proyecto' => $proyecto,
        'form' => $form->createView(),
    ));
}
```

Figura 9. *Uso del patrón Método de Fábrica. (Fuente: Elaboración propia)*

- **Estructurales**

Adaptador (del inglés Adapter): permite a una interfaz de una clase existente ser usada por otra interfaz; y a las clases, trabajar con otras sin cambiar su código. La entidad Usuario es un componente del núcleo de Seguridad. Esta le permite implementar los métodos getUsername(), setPassword() necesarias para el correcto funcionamiento de la autenticación en Symfony3.

```php
public function getUsername()
{
    return $this->username;
}

/** Set password ...*/
public function setPassword($password)
{
    $this->password = $password;

    return $this;
}
```

***Figura 10**.Uso del patrón Adaptador en la clase Usuario.php. (Fuente: Elaboración propia)*

Decorador (del inglés Decorator): aplicado a la generación de vistas, la solución que ofrece este patrón es la de añadir funcionalidad adicional a las plantillas. Ejemplo, añadir el menú y el pie de página a las plantillas que lo requieran; se trata de decorar las plantillas con elementos adicionales reutilizables. El sistema de plantillas Twig está provisto de un mecanismo de herencia gracias al cual la decoración de plantillas resulta de una flexibilidad y versatilidad total.

```
    <i class="m-menu__ver-arrow la la-angle-right"></i>
</a>
<div class="m-menu__submenu ">
    <span class="m-menu__arrow"></span>
    <ul class="m-menu__subnav">
        <li class="m-menu__item  m-menu__item--parent" aria-haspopup="true"  data-redirect="true">
                    <span class="m-menu__link">
                        <span class="m-menu__link-text">
                            Gestionar
                        </span>
                    </span>
        </li>
        <li class="m-menu__item " aria-haspopup="true"  data-redirect="true">
            <a  href="{{ path('profesor_index') }}" class="m-menu__link ">
                <i class="m-menu__link-icon la la-user"></i>
                <span class="m-menu__link-text">
                            Profesores
                        </span>
            </a>
        </li>
        <li class="m-menu__item " aria-haspopup="true"  data-redirect="true">
            <a  href="{{ path('estudiante_index') }}" class="m-menu__link ">
                <i class="m-menu__link-icon la  la-graduation-cap"></i>
                <span class="m-menu__link-text">
                            Estudiantes
                        </span>
            </a>
        </li>
        <li class="m-menu__item " aria-haspopup="true"  data-redirect="true">
            <a  href="inner.html" class="m-menu__link ">
                <i class="m-menu__link-icon la  la-rocket"></i>
```

***Figura 11.** Uso del patrón Decorador. (Fuente: Elaboración propia)*

- **Comportamiento**

Mediador (del inglés Mediator): Las clases controladoras son mediadoras entre las capas de modelo y vista.

2.5 Validación del diseño

Para la validación del diseño se aplicaron las métricas Tamaño Operacional de Clases (TOC) y Relaciones entre Clases (RC).

2.5.1 Tamaño Operacional de Clases (TOC)

Los resultados obtenidos luego de aplicar la métrica se muestran en la Figura 12.

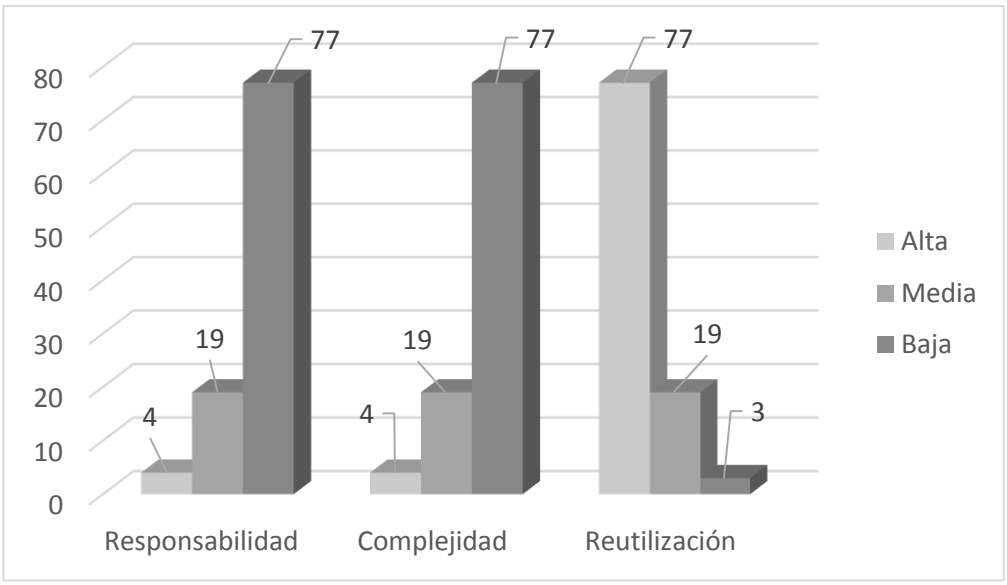


Figura 12. *Representación de los atributos de calidad de la métrica TOC. (Fuente: Elaboración propia)*

2.5.2 Relaciones entre Clases (RC)

Los resultados obtenidos luego de aplicar la métrica se muestran en la Ilustración 13.

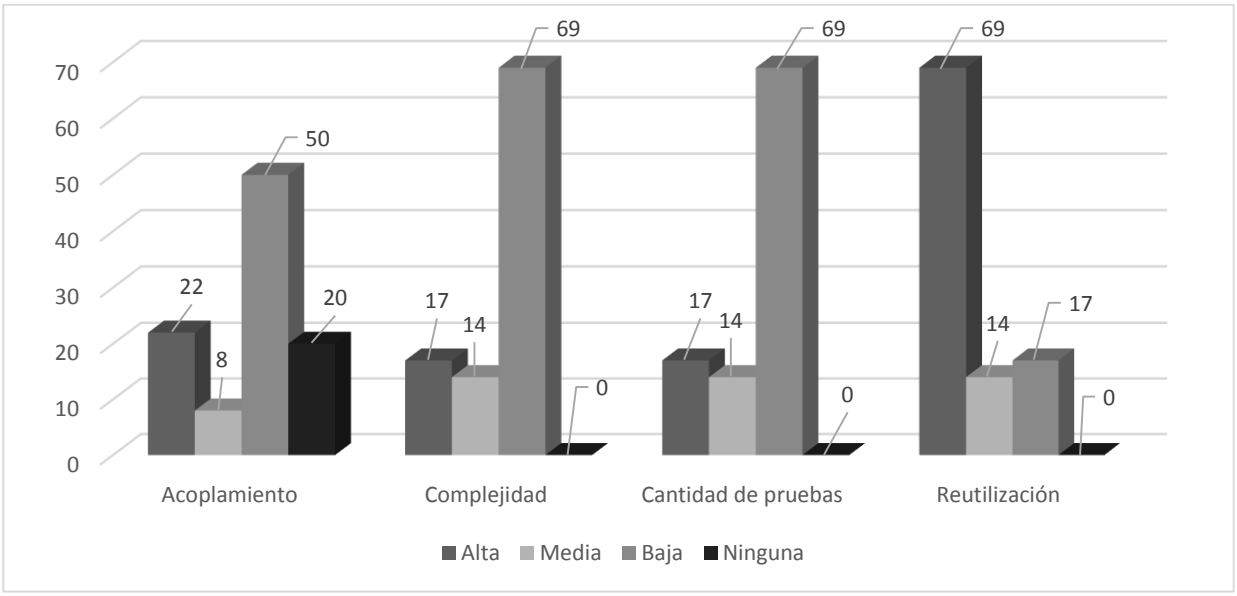


Figura 13.*Representación de los atributos de calidad de la métrica RC. (Fuente: Elaboración propia)*

Luego de aplicadas las métricas se verificó que el diseño propuesto provee un nivel bajo de responsabilidad. Esto trae consigo un porciento bajo de complejidad de implementación, lo cual significa un menor grado de dificultad en el desarrollo de la herramienta. El 69% de las clases presentan un nivel bajo de mantenimiento, lo que reduce el grado de esfuerzo necesario para desarrollar un arreglo o rectificar algún error. Teniendo en cuenta que el 69% de las clases tienen una baja necesidad de realizar pruebas unitarias al sistema diseñado, se reduce el esfuerzo necesario para la realización de estas. En cuanto a la reutilización se alcanzaron los resultados deseados, pues se obtuvieron valores altos de reutilización en ambas métricas, 77% en TOC y 69% en RC, ya que se busca el poder reutilizar las clases en

caso de ser necesario. Se tiene en cuenta además el acoplamiento, que atendiendo a los resultados obtenidos el 50% de las clases tienen bajo acoplamiento, lo que evidencia el uso del patrón bajo acoplamiento. Con los resultados anteriores queda validado el diseño de la herramienta a desarrollar, atendiendo a los parámetros definidos.

2.6 Conclusiones parciales

Al finalizar el presente capítulo se arribó a las siguientes conclusiones parciales:

- Se confeccionó el modelo conceptual para una representación visual del dominio del problema.
- La confección de los artefactos historias de usuario, plan de iteraciones, tarjetas CRC, y el modelo de datos permitieron organizar el proceso de desarrollo del sistema.
- Se fundamentó la elección del patrón Modelo Vista Controlador para la arquitectura del sistema mediante un diagrama de componentes.
- El uso de los patrones de diseño y de las métricas TOC y RC para la validación de este contribuyeron a la obtención de un diseño con calidad.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

El presente capítulo, tiene como objetivo validar la propuesta de solución aplicando las pruebas de validación de software abordadas en el capítulo 1 (epígrafe 1.12). Se muestran las tareas de ingeniería, artefactos generados en la fase de desarrollo de la metodología de software utilizada. Se abordan los estándares de codificación empleados durante la implementación del sistema para garantizar un buen entendimiento y legibilidad del código. Se desarrolla un caso de estudio empleando datos reales provenientes de profesores del departamento de Ingeniería y Gestión de Software. Se describen en detalle los principales resultados obtenidos con el caso de estudio.

3.2 Fase de desarrollo

En la fase de desarrollo se planifican y ejecutan las tareas de ingeniería. Durante esta se codifica todo el sistema diseñado y se obtiene como resultado la herramienta propuesta.

3.2.1 Tareas de ingeniería por iteraciones

Las tareas de ingeniería son actividades que se derivan de las HU para simplificar su implementación. A continuación, se muestran las tareas a desarrollar para la implementación, por cada HU.

Tabla 10. *Tareas ingenieriles por iteraciones. (Fuente: Elaboración propia)*

Iteración	Historia de usuario	Tareas de ingeniería
1	Adicionar estudiante	Implementar una funcionalidad que permita al profesor principal o al administrador adicionar un estudiante en el sistema.
	Modificar estudiante	Implementar una funcionalidad que permita al profesor principal o al administrador modificar un estudiante del sistema.
	Eliminar estudiante	Implementar una funcionalidad que permita al profesor principal o al administrador eliminar un estudiante del sistema.
	Adicionar proyecto de curso	Implementar una funcionalidad que permita al profesor principal o al administrador adicionar un proyecto de curso en el sistema.
	Modificar proyecto de curso	Implementar una funcionalidad que permita al profesor principal o al administrador modificar un proyecto de curso del sistema.

	Eliminar proyecto de curso	Implementar una funcionalidad que permita al profesor principal o al administrador eliminar un proyecto de curso del sistema.
	Modificar escala de Saaty	Implementar una funcionalidad que permita al sistema asignar la prioridad de los criterios usando escala de Saaty.
	Ordenar criterio	Implementar una funcionalidad que ordene los criterios en el sistema.
	Combinar valoraciones	Implementar una funcionalidad que combine las valoraciones en el sistema.
	Evaluar proyectos	Implementar una funcionalidad que permita al profesor principal o a los profesores evaluar los proyectos en el sistema.
	Adicionar objetivos de la asignatura	Implementar una funcionalidad que permita al profesor principal adicionar los objetivos de la asignatura en el sistema.
	Unificar evaluación a un mismo dominio	Implementar una funcionalidad que unifique las evaluaciones en un mismo dominio.
	Transformar evaluación a 2-Tuplas	Implementar una funcionalidad que permita transformas las evaluaciones a 2-tuplas.
	Agregar evaluación global de cada proyecto	Implementar una funcionalidad que permita agregar una evaluación global de cada estudiante.
	Calcular valor colectivo de cada criterio para cada estudiante	Implementar una funcionalidad que permita calcular el valor colectivo de cada criterio para cada estudiante mediante el operador Media Aritmética Extendida.
	Agregar valor de los criterios para cada estudiante	Implementar una funcionalidad que permita agregar el valor de los criterios para cada estudiante el operador Media Ponderada Extendida.
	Adicionar evaluación de estudiante	Implementar una funcionalidad que permita al profesor adicionar la evaluación del estudiante en el sistema.

	Generar ordenado de proyectos evaluados	Implementar una funcionalidad que permita generar un listado ordenado de proyectos evaluados.
	Mostrar resultados de la evaluación	Implementar una funcionalidad que permita exportar los resultados de la evaluación.
	Adicionar profesor a un proyecto	Implementar una funcionalidad que permita al profesor principal adicionar un profesor al proyecto en el sistema.
	Adicionar estudiante a un proyecto	Implementar una funcionalidad que permita al profesor principal adicionar un estudiante a un proyecto.
	Adicionar criterio a un proyecto	Implementar una funcionalidad que permita al profesor principal adicionar criterios a un proyecto
2	Adicionar profesor	Implementar una funcionalidad que permita al profesor principal o al administrador adicionar un profesor en el sistema.
	Modificar profesor	Implementar una funcionalidad que permita al profesor principal o al administrador modificar un profesor del sistema.
	Eliminar profesor	Implementar una funcionalidad que permita al profesor principal o al administrador eliminar un profesor del sistema.
	Listar profesor	Implementar una funcionalidad que permita al profesor principal listar los profesores en el sistema.
	Buscar profesor	Implementar una funcionalidad que permita al profesor principal o al administrador buscar un profesor en el sistema.
	Mostrar profesor	Implementar una funcionalidad que permita mostrar un profesor.
	Modificar objetivos de la asignatura	Implementar una funcionalidad que permita al profesor principal o a l administrador modificar los objetivos de la asignatura en el sistema.

	Eliminar objetivos de la asignatura	Implementar una funcionalidad que permita al profesor principal o al administrador eliminar los objetivos de la asignatura en el sistema.
	Ordenar evaluación de cada estudiante	Implementar una funcionalidad que permita ordenar las evaluaciones de los estudiantes.
	Adicionar rol	Implementar una funcionalidad que permita al administrador adicionar un rol en el sistema.
	Modificar rol	Implementar una funcionalidad que permita al administrador modificar un rol en el sistema.
	Eliminar rol	Implementar una funcionalidad que permita al administrador eliminar un rol del sistema.
	Eliminar evaluación de estudiante	Implementar una funcionalidad que permita al profesor eliminar una evaluación del estudiante en el sistema.
	Modificar evaluación de estudiante	Implementar una funcionalidad que permita al profesor modificar una evaluación del estudiante en el sistema.
	Mostrar resultados de la evaluación	Implementar una funcionalidad que permita mostrar los resultados de la evaluación.
	Adicionar Nomenclador Asignatura	Implementar una funcionalidad que permita al administrador adicionar nomenclador asignatura en el sistema.
	Adicionar Nomenclador Categoría docente	Implementar una funcionalidad que permita al administrador adicionar nomenclador categoría docente en el sistema.
	Adicionar Nomenclador Grado científico	Implementar una funcionalidad que permita al administrador adicionar nomenclador grado científico en el sistema.
	Modificar Nomenclador Asignatura	Implementar una funcionalidad que permita al administrador modificar nomenclador asignatura en el sistema.
	Modificar Nomenclador Categoría docente	Implementar una funcionalidad que permita al administrador modificar nomenclador categoría docente en el sistema.

	Modificar Nomenclador Grado científico	Implementar una funcionalidad que permita al administrador modificar nomenclador grado científico en el sistema.
	Mostrar Nomenclador Asignatura	Implementar una funcionalidad que permita al administrador Mostrar nomenclador asignatura en el sistema.
	Mostrar Nomenclador Categoría docente	Implementar una funcionalidad que permita al administrador Mostrar nomenclador categoría docente en el sistema.
	Mostrar Nomenclador Grado científico	Implementar una funcionalidad que permita al administrador Mostrar nomenclador grado científico en el sistema.
	Agregar Dominio	Implementar una funcionalidad que permita al administrador agregar un dominio en el sistema.
3	Listar estudiante	Implementar una funcionalidad que permita listar los estudiantes.
	Buscar estudiante	Implementar una funcionalidad que permita buscar un estudiante en el sistema.
	Listar proyecto de curso	Implementar una funcionalidad que permita listar los proyectos de curso.
	Buscar proyecto de curso	Implementar una funcionalidad que permita buscar proyectos de curso.
	Listar objetivos de la asignatura	Implementar una funcionalidad que permita listar objetivos de la asignatura.
	Buscar objetivos de la asignatura	Implementar una funcionalidad que permita buscar los objetivos de la asignatura en el sistema.
	Listar rol	Implementar una funcionalidad que permita listar los roles en el sistema.
	Buscar rol	Implementar una funcionalidad que permita buscar un rol en el sistema.
	Listar evaluaciones de estudiantes	Implementar una funcionalidad que permita listar las evaluaciones de los estudiantes.

	Buscar evaluaciones de estudiantes	Implementar una funcionalidad que permita buscar las evaluaciones de los estudiantes en el sistema.
	Eliminar Nomenclador Asignatura	Implementar una funcionalidad que permita al administrador eliminar nomenclador asignatura del sistema.
	Eliminar Nomenclador Categoría docente	Implementar una funcionalidad que permita al administrador eliminar nomenclador categoría docente del sistema.
	Eliminar Nomenclador Grado científico	Implementar una funcionalidad que permita al administrador eliminar nomenclador grado científico del sistema.
	Listar Nomenclador Asignatura	Implementar una funcionalidad que permita al administrador listar nomenclador asignatura en el sistema.
	Listar Nomenclador Categoría docente	Implementar una funcionalidad que permita al administrador listar nomenclador categoría docente en el sistema.
	Listar Nomenclador Grado científico	Implementar una funcionalidad que permita al administrador listar nomenclador grado científico en el sistema.
	Buscar Nomenclador Asignatura	Implementar una funcionalidad que permita al administrador buscar nomenclador asignatura en el sistema.
	Buscar Nomenclador Categoría docente	Implementar una funcionalidad que permita al administrador buscar nomenclador categoría docente en el sistema.
	Buscar Nomenclador Grado científico	Implementar una funcionalidad que permita al administrador buscar nomenclador grado científico en el sistema.
	Eliminar profesor de un proyecto	Implementar una funcionalidad que permita al profesor principal eliminar un profesor de un proyecto del sistema.

	Eliminar estudiante de un proyecto	Implementar una funcionalidad que permita al profesor principal eliminar un estudiante de un proyecto del sistema.
	Eliminar criterio de un proyecto	Implementar una funcionalidad que permita al profesor principal eliminar un criterio de un proyecto del sistema.

3.2.2 Tareas detalladas

A continuación, se muestran las tareas de ingeniería detalladas correspondientes a las historias de usuario Adicionar proyecto de curso el resto de las tareas detalladas se encuentran en el Anexo 5.

Tabla 11*.Tarea de ingeniería de la historia de usuario: "Adicionar Proyecto de Curso". (Fuente: Elaboración propia)*

TAREA DE INGENIERÍA	
Número tarea: 4	Historia de usuario: Adicionar Proyecto de Curso
Nombre tarea: Adicionar Proyecto de Curso	
Tipo de Tarea: Desarrollo (Desarrollo, Corrección, Mejora)	Puntos estimados (días): 3
Fecha inicio: 16/01/2018	Fecha fin: 19/01/2018
Programador responsable: Jenniffer Maulini	
Descripción: Implementar una funcionalidad que permita al profesor principal o al administrador adicionar un proyecto de curso en el sistema.	

3.2.3 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. (Google, 2015)

En función de lograr estandarización en el código del sistema, se definen un conjunto de pautas a seguir durante la implementación.

Definición de clases

Las clases se encuentran en archivos independientes que solo contendrán el código de esta. Se utiliza el estilo de codificación “UpperCamelCase”, el cual establece que los nombres inician con letra mayúscula y si poseen más de una palabra, la primera letra de estas deberá ser mayúsculas también.

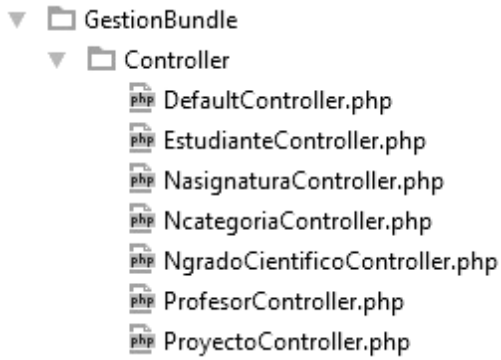


Figura 14. *Definición de clases. (Fuente: Elaboración propia)*

Definición de variables

Los nombres de las variables deben ser descriptivos y concisos. No se usan abreviaciones. Se utiliza el estilo de codificación “lowerCamelCase”, el cual establece que los nombres inician con letra minúscula y cada nueva palabra debe iniciar con mayúscula.

```
/**
 * @return int
 */
public function getId()
{
    return $this->id;
}

/**
 * @param int $id
 */
public function setId($id)
{
    $this->id = $id;
}
```

FIGURA 15. DEFINICIÓN DE VARIABLES. (FUENTE: ELABORACIÓN PROPIA)

3.3 Fase de pruebas

Uno de los pilares de la metodología X.P es el uso de test para comprobar el funcionamiento de los códigos que vayamos implementando. Este proceso se realiza de forma continua para asegurar durante todo el proceso de desarrollo, el éxito del producto. Esto permite detectar los errores en un plazo de tiempo corto.

3.3.1 Pruebas de caja blanca

El uso de esta técnica es mostrado en el siguiente ejemplo. Se analizan y enumeran las sentencias de código del método TranformarDosTuplas contenidas en la clase TowTuples. Este método transforma las evaluaciones emitidas por los profesores en los dominios que estimen convenientes en un mismo dominio, en este caso el lingüístico. Ver la Figura 15.

```
private function tranformarDosTuplas($evaluacionesp)
{
    $data = array();                                                    //1
    foreach ($evaluacionesp as $evaluacion) {                           //2
        $aux = array();                                                 //3
        foreach ($evaluacion as $eval) {                                //4
            $da = array();                                              //5
            if ($eval instanceof towTuplesDominioIntervalo)             //6
                $da = $this->dominioLinguisticoG->transformarIntervalar($eval->getA(), $eval->getB(), $eval->getMax()); //7
            elseif ($eval instanceof towTuplesDominioNumerico)                                                          //8
                $da = $this->dominioLinguisticoG->transformarNumerico($eval->getIndice(), $eval->getMax());             //9
            elseif ($eval instanceof towTuplesDominioLinguistico)                                                       //10
                $da = $this->dominioLinguisticoG->transformarLinguistico($this->dominioLinguisticoP, $eval->getIndex()); //11
            $aux[] = $da;                                                                                               //12
        }
        $data[] = $aux;                                                                                                 //13

    }
    return $data;                                                                                                       //14

}
```

***Figura 16**. Código del método TranformarDosTuplas (). (Fuente: Elaboración poropia)*

Representar el procedimiento en un grafo de flujo

A continuación, se muestra el grafo de flujo correspondiente al código del método seleccionado:

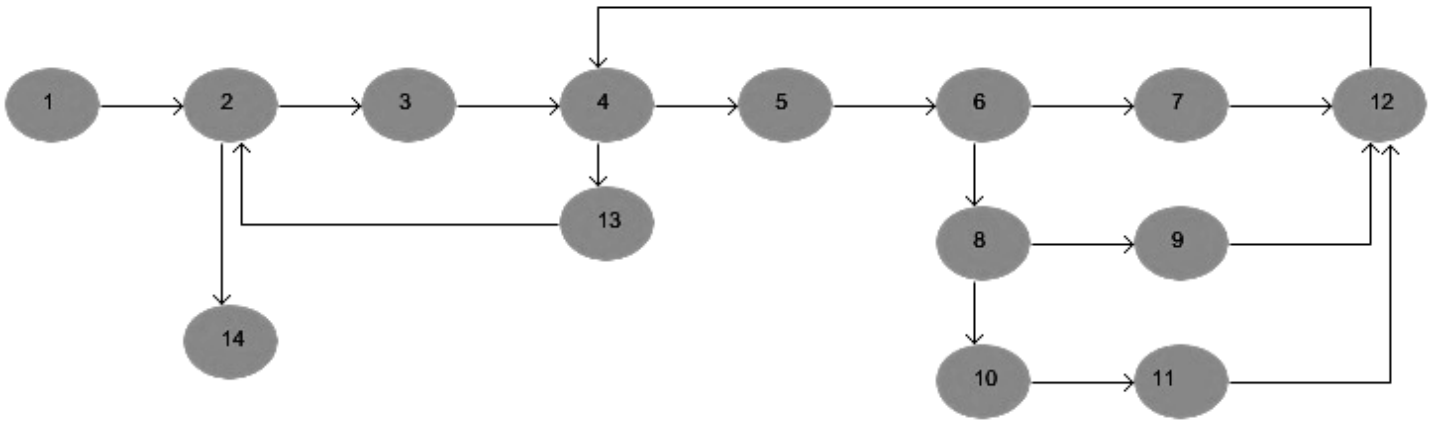


***Figura 17.** Grafo de flujo. (Fuente: Elaboración propia)*

Calcular la complejidad ciclomática:

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, la cual es una métrica de software útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa (Pressman, 2010). El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar.

V(G) = Aristas – Nodos + 2 V(G) = Nodos Predicado + 1

A: número de aristas N: número de nodos.

V(G)= (17 - 14) + 2 = 5

P: número de nodos predicados (nodos de los cuales parten dos o más aristas).

V(G)= 4 + 1 = 5

V(G)= R

R: número de regiones del grafo.

V(G)= 5

Luego de calculada la complejidad ciclomática mediante las tres fórmulas se evidencia que la complejidad es 5, de manera que existen 5 posibles caminos por donde el flujo puede circular. Este valor representa el número mínimo de casos de pruebas para el procedimiento tratado. Seguidamente, es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución.

Conjunto básico de caminos independientes

Los caminos independientes resultantes son:

Camino 1: 1-2-14.

Camino 2: 1-2-3-4-13-2-14.

Camino 3: 1-2-3-4-5-6-7-12-4-13-2-14.

Camino 4: 1-2-3-4-5-6-8-9-12-4-13-2-14.

Camino 5: 1-2-3-4-5-6-8-10-11-12-4-13-2-14.

Se ejecutan los casos de prueba para cada camino independiente determinado en el grafo de flujo.

***Tabla 12**. Caso de prueba Camino 1. (Fuente: Elaboración propia)*

CASO DE PRUEBA CAMINO 1	
Condición de ejecución	No es introducida al menos una evaluación.
Entrada	Evaluaciones en dominio numérico, intervalar y lingüístico.
Resultados esperados	Se muestra un mensaje de error informando al usuario que debe existir al menos una evaluación.

***Tabla 133**. Caso de prueba Camino 1. (Fuente: Elaboración propia)*

CASO DE PRUEBA CAMINO 2	
Condición de ejecución	No existe al menos un criterio.
Entrada	Evaluaciones en dominio numérico, intervalar y lingüístico.

Resultados esperados	Se muestra un mensaje de error informando al usuario que debe existir al menos un criterio.

Tabla 14.*Caso de prueba Camino 3 (Fuente: Elaboración propia)*

CASO DE PRUEBA CAMINO 3	
Condición de ejecución	Se introducen evaluaciones de tipo intervalar.
Entrada	Evaluaciones en dominio numérico, intervalar y lingüístico.
Resultados esperados	Se trasforman las evaluaciones a 2-Tuplas y seguidamente crea un arreglo con las mismas.

Tabla 15*. Caso de prueba Camino 4(Fuente: Elaboración propia)*

CASO DE PRUEBA CAMINO 4	
Condición de ejecución	Se introducen evaluaciones de tipo numérico.
Entrada	Evaluaciones en dominio numérico, intervalar y lingüístico.
Resultados esperados	Se trasforman las evaluaciones a 2-Tuplas y seguidamente crea un arreglo con las mismas.

Tabla 16*. Caso de prueba Camino 5. (Fuente: Elaboración propia)*

CASO DE PRUEBA CAMINO 5	
Condición de ejecución	Se introducen evaluaciones de tipo lingüístico.
Entrada	Evaluaciones en dominio numérico, intervalar y lingüístico.
Resultados esperados	Se trasforman las evaluaciones a 2-Tuplas y seguidamente crea un arreglo con las mismas.

Análisis de los resultados:

Para la validación del código generado en el desarrollo de la herramienta se seleccionó uno de los principales métodos del sistema. A este se le realizó las pruebas para evaluar si el funcionamiento del mismo se comportó de la manera esperada. Las pruebas realizadas a esta funcionalidad resultaron satisfactorias, comprobándose la estabilidad de la lógica aplicada en el código.

3.3.2 Pruebas de caja negra.

Las pruebas de caja negra se centran en los requisitos funcionales del sistema. Con estas pruebas se intentan encontrar funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en acceso a base de datos externas y errores de rendimiento. Se centran en qué hace el software y no en cómo lo hace. (Pressman, 2010)

Según (Pressman, 2010) existen varias técnicas para realizar este tipo de pruebas, se seleccionó la técnica partición equivalente, la cual permite comprobar los valores válidos e inválidos de las entradas existentes en la aplicación.

Partición equivalente: método que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Se muestra seguidamente un ejemplo de casos de prueba para la herramienta:

***Tabla 17**. Caso de prueba de Gestionar proyecto de curso. (Fuente: Elaboración propia)*

Escenario	**Descripción**	**Nombre**	**Estudiantes**	**Profesores**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Adicionar nuevo proyecto	Permite crear un nuevo proyecto. Todos los campos son obligatorios.	V Proyecto I	V Amanda González Abella Annet Herrero González	V Marieta Peña Abreu Yadira García García	El sistema muestra el mensaje de confirmación: "¡Proyecto agregado con éxito!" El sistema cierra la interfaz. Se actualiza la lista de proyectos.	Seleccionar del menú lateral izquierdo la opción "Gestionar". Seleccionar la opción "Proyecto". Seleccionar la opción "Adicionar". Se muestra una pantalla con una secuencia de pasos a seguir. Se deben llenar los campos que aparecen en cada uno de ellos. Una vez introducidos los valores se muestra una interfaz con los valores añadidos, seleccionar la opción "Guardar". Se actualiza el listado de los proyectos.
		I 	V Amanda González Abella Annet Herrero González	V Marieta Peña Abreu Yadira García García	El sistema muestra un mensaje mostrando que existen campos erróneos. Se muestra el mensaje de error en rojo debajo del campo con valores incorrectos o vacíos: "¡debes	

					de especificar el nombre del proyecto!" o "Introduzca solo letras, por favor"	
		V	I	V	El sistema muestra un mensaje mostrando que existen campos erróneos. Se muestra el mensaje de error en rojo debajo del campo con valores incorrectos o vacíos: "¡agrega al menos un estudiante!"	
		Proyecto I		Marieta Peña Abreu Yadira García García		
		V	V	I	El sistema muestra un mensaje mostrando que existen campos erróneos. Se muestra el mensaje de error en rojo debajo del campo con valores incorrectos o vacíos:	
		Proyecto I	Amanda González Abella Annet Herrero González			

					"¡agrega al menos un profesor!"	
EC 1.2 Modificar proyecto	Permite modificar valores de los proyectos existentes	V Proyecto I	V Amanda González Abella Annet Herrero González	V Marieta Peña Abreu Yadira García García	El sistema muestra el mensaje de confirmación: "¡Proyecto agregado con éxito!" El sistema cierra la interfaz. Se actualiza la lista de proyectos.	Ejecutar el flujo central del EC 1.1(1,2). Seleccionar el botón "Editar" a la derecha del proyecto que se desea modificar. Una vez introducidos los datos que se quieren modificar seleccionar entonces la opción "Guardar". Se actualiza el listado de los proyectos.
		I V	V Amanda González Abella Annet Herrero González I	V Marieta Peña Abreu Yadira García García V	El sistema muestra un mensaje mostrando que existen campos erróneos. Se muestra el mensaje de error en rojo debajo del campo con valores incorrectos o vacíos: "¡debes de especificar el nombre del proyecto!" o "¡Introduzca solo letras, por favor"	

		Proyecto I		Marieta Peña Abreu Yadira García García	El sistema muestra un mensaje mostrando que existen campos erróneos. Se muestra el mensaje de error en rojo debajo del campo con valores incorrectos o vacíos: "¡agrega al menos un estudiante!"	
		V	V	I	El sistema muestra un mensaje mostrando que existen campos erróneos. Se muestra el mensaje de error en rojo debajo del campo con valores incorrectos o vacíos: "¡agrega al menos un profesor!"	
		Proyecto I	Amanda González Abella Annet Herrero González			
EC 1.3 Eliminar proyecto	Permite eliminar	V	V	V	El sistema muestra el mensaje	Ejecutar el flujo central del EC 1.1(1,2)
		Proyecto I	1. Amanda González Abella	1. Marieta Peña Abreu		

	proyectos de curso		Annet Herrero González	Yadira García García	"Eliminado. El elemento se Ha eliminado"	Seleccionar el botón "Eliminar" a la derecha del proyecto que desea eliminar. El sistema muestra el mensaje: "¿Seguro que desea eliminar el elemento? ¡no podrás revertir los cambios!", seguido de dos opciones: "¡No. Cancelar!" y "Sí. ¡Eliminar!". Seleccionar la opción "Si. ¡Eliminar!". El sistema muestra el mensaje de confirmación: ""Eliminado. El elemento se Ha eliminado". Se actualiza la lista de proyectos.

Análisis de los resultados

Se realizaron dos iteraciones y un caso de prueba para cada requisito. En la primera iteración se detectaron 19 no conformidades, siendo 4de Validación, 4 de Correspondencia, 5 de Visibilidad, 3 de Error de idioma, 2 de Ortografía y 1 de Redacción. En una segunda iteración estas no conformidades fueron solucionadas en su totalidad.

3.3.3 Caso de estudio.

Con el objetivo de validar la solución al problema de investigación se diseña un caso de estudio. Se aplica el método durante la evaluación de los proyectos de curso de la asignatura Ingeniería de Software 1 en la facultad 3 de la Universidad de Ciencias Informáticas. Se evaluó el desempeño de 9 estudiantes, quienes integraron 3 equipos en los que desarrollaron sus proyectos de curso. Participaron en el tribunal de evaluación cinco profesores. Para el caso de estudio se cuenta con un equipo de cómputo con las siguientes prestaciones:

Tipo de CPU: Intel (R) Celeron (R) a 2.16GHz

Memoria del sistema: 4GB RAM DDR3 SDRAM

Diseño experimental

Para realizar el proceso de evaluación se deben seguir una serie de pasos, los cuales se exponen a continuación:

1 El profesor principal de la asignatura debe añadir al sistema los proyectos a evaluar. Para esto debe introducir el título de los proyectos que se van a evaluar, luego debe seleccionar los estudiantes que componen los mismos, así como los profesores que conforman el claustro de evaluadores. Como se muestra en las tablas 17 y 18.

#	Nombre	Asignatura	Categoría
1	Ailia Parra Fernandez	Ingienería de Software I	Profesor Asistente
2	Carlos Javier Perez de la Cruz	Ingienería de Software I	Profesor Auxiliar
3	Carlos Rafael Rodríguez Rodríguez	Ingienería de Software I	Profesor Titular

***Figura 18**. Claustro de profesores. (Fuente: Elaboración propia)*

***Figura 19**.Proyecto de curso I y estudiantes que lo conforman. (Fuente: Elaboración propia)*

2 Luego de creado el proyecto el profesor principal adiciona los criterios que se tendrán en cuenta para evaluar el desempeño de los estudiantes a partir de los objetivos previstos en el plan de estudios y en el programa analítico de la asignatura. Luego asigna pesos a los mismos, para esto se establece una comparación entre ellos según la importancia que tienen unos respecto a otros, como se muestra a continuación.

#	Nombre	Tipo	Peso	Acciones
1	Desarrollo del pensamiento lógico, capacidad de abstracción y razonamiento mediante la modelación	Idividual	0.53839383973865	
2	Modelado de la estructura	Colectivo	0.28335444362341	
3	Modelado del comportamiento	Colectivo	0.1180880848349	
4	Aplicación de metodología de desarrollo de software y una herramienta CASE con enfoque sistémico de manera correcta	Idividual	0.060163631803037	

Figura 20. *Ejemplo de algunos criterios considerados en la evaluación y pesos de los mismos. (Fuente: Elaboración propia)*

3 Una vez realizados estos pasos el profesor principal está listo para cambiar el estado del proyecto a "Someter evaluación", y es entonces cuando el resto de profesores pueden evaluar los estudiantes.

4 Los profesores que conforman el claustro evalúan los estudiantes de cada proyecto, basándose en los criterios establecidos previamente. Esta evaluación la emiten en el dominio de expresión de su preferencia. Para esto los profesores deben seguir los siguientes pasos.

1) En la interfaz "Mis evaluaciones" se muestran los proyectos a los cuales pertenece y en el estado que están los mismos ya sea “pendiente”, “en evaluación” o “evaluado”.

2) Escoge el proyecto que tenga el estado "en evaluación" y realiza la misma a cada estudiante.

3) Dentro de cada estudiante aparece una interfaz que muestra el nombre del mismo, el grupo al que pertenece y los criterios a evaluar. Debe emitir la evaluación en el dominio de expresión que estime conveniente y hecho esto da en la opción terminar.

5 Cuando los profesores terminan de realizar la evaluación el sistema unifica todas las notas en el dominio número y la lleva a 2-tuplas. Los estudiantes de cada proyecto tendrán al lado de su nombre esta evaluación.

6 Una vez culminada la evaluación de todos los profesores se realiza la agregación global de cada estudiante obteniendo una nota general de acuerdo a las evaluaciones emitidas por los evaluadores, tal como se muestra en la Ilustración 18.

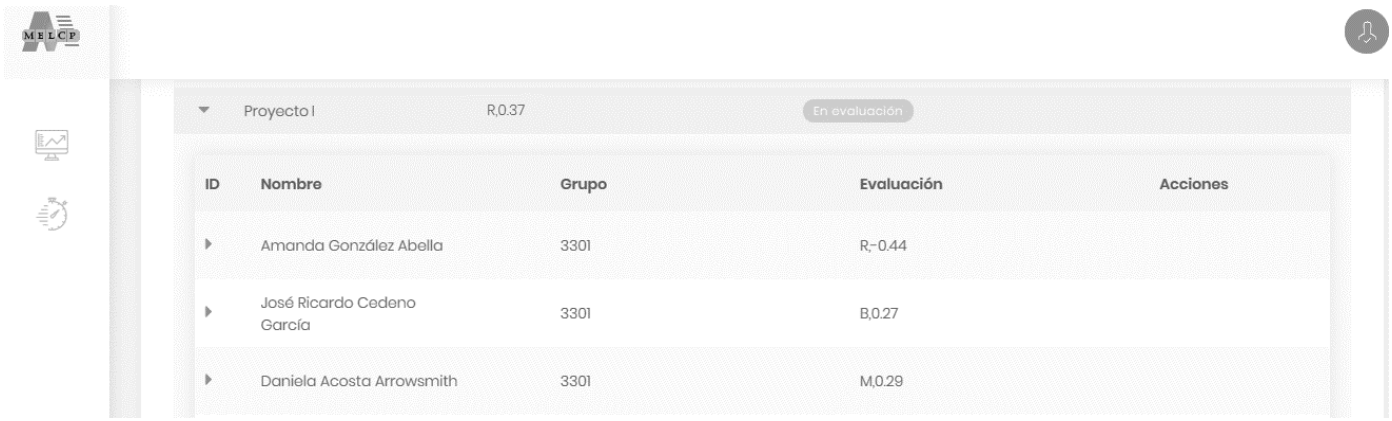


***Figura 21**.Evaluación de estudiantes del proyecto "Proyecto I". (Fuente: Elaboración propia)*

7 Al terminar esta serie de pasos el profesor principal puede visualizar la evaluación general de cada proyecto, y de los estudiantes que lo conforman, así como la evaluación que emitió cada profesor a cada estudiante en cada uno de los criterios. Teniendo la evaluación en el dominio que fue evaluado y su equivalente en 2-tuplas. En la siguiente imagen se evidencian las evaluaciones de un criterio para la estudiante Amanda González Abella por todos los profesores.

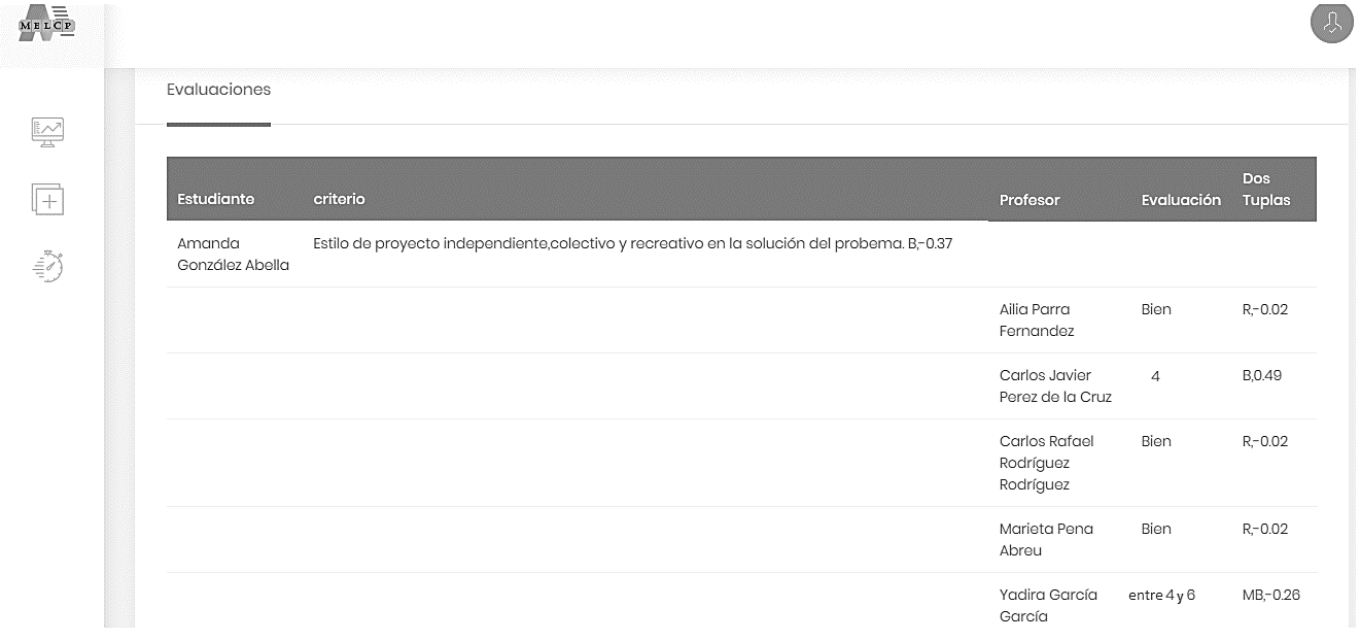


***Figura 22**. Evaluaciones de todos los profesores según los criterios. (Fuente: Elaboración propia)*

Análisis de resultados

Con la realización de este proceso se obtuvo como resultados:

- Los profesores obtienen la evaluación global de cada equipo (expresada en 2-tuplas).
- La evaluación integrada de los estudiantes (expresada en 2-tuplas) y su orden dentro del equipo

- Realiza tratamiento a la variabilidad de los datos, la imprecisión y la vaguedad como fuentes de incertidumbre en la información, presentes en el proceso de evaluación del aprendizaje a partir de la computación con palabras.

Los resultados anteriormente mencionados ayudan a los profesores en la toma de decisiones para la evaluación final del estudiante. La posibilidad de tener una evaluación basada en 2-tuplas les permite ser más flexibles con las notas que emiten a los mismos. También les ayuda a tener un mejor acercamiento hacia los estudiantes ya que pueden apreciar el grado de cumplimiento de los objetivos de la asignatura, lo que les permite tener una atención diferenciada con cada uno de ellos dependiendo de los objetivos que no hayan dominado por completo. Si no se llevara a cabo este método podrían existir diferentes estudiantes con una misma nota cuantitativa, sin embargo, la utilización de éste permite diferenciar el nivel de aprendizaje de los estudiantes logando emitir una nota más fiable acorde a las características de los mismos.

3.3.4 Pruebas de aceptación

Es la disciplina que realizará la prueba final antes del despliegue del módulo. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (Rodriguez, 2014).

Las Pruebas de aceptación son realizadas por el cliente verificando que el módulo cumple con los requerimientos especificados para validar las variables por las que se decidió iniciar la investigación.

Validación de las variables de la investigación

A continuación, se explica cómo se evalúa la variable interpretabilidad.

Antes de conocer el resultado de la validación de las variables es necesario definir el concepto de interpretabilidad.

La interpretabilidad: refleja la facilidad con que el usuario puede entender los datos, utilizarlos y analizarlos correctamente.

Tabla 18.*Resultado de la validación de la variable interpretabilidad. (Fuente: Elaboración propia)*

Variables	**Dimensiones**	**Proceso actual**	**Proceso con la utilización de la herramienta implementada**
Interpretabilidad de los resultados.	Evaluación de trabajos de curso	En la evaluación participan varios profesores expresando su criterio en un solo dominio de expresión, el dominio numérico. Esta información cuantitativa es determinista pero no siempre representa la información cualitativa de manera precisa.	Los profesores pueden emitir su evaluación en diferentes dominios de expresión, dominio numérico, lingüístico e intervalar, según estimen conveniente.

		No se tiene en cuenta la posibilidad de que profesores emitan notas en diferentes dominios de expresión, lo cual trae consigo una nota poco flexible a la hora de evaluar un estudiante, ya que una evaluación numérica no es capaz de describir la capacidad del mismo en cuanto a conocimientos adquiridos. Así como poca interpretabilidad de los resultados.	El sistema tiene como resultado los valores colectivos de los criterios y la evaluación final para cada estudiante expresada en 2-tuplas, donde el primer valor representa la información lingüística del estudiante, y el segundo la traslación simbólica, la cual representa el nivel de importancia de esa información lingüística. Dando una mejor interpretación de la información obtenida.
		Se evalúan muchos criterios los cuales miden el cumplimiento de los objetivos de la asignatura evaluada, así como su integración con otras asignaturas, lo cual puede resultar engorroso a la hora de evaluar, teniendo pérdida de información y lentitud en el proceso.	La evaluación por criterios ayuda al profesor para un mejor acercamiento con el estudiante, logrando redefinir el proceso de aprendizaje del mismo a partir de la evaluación dada.

Para calcular el grado de satisfacción del cliente con la herramienta realizada, se aplicó la técnica Iadov. Durante la valoración participó como cliente, la DrC.Marieta Peña Abreu y a tres profesores del departamento de ISGW.

Técnica Iadov

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

1 Máxima satisfacción.

2 Más satisfecho que insatisfecho.

3 No definida.

4 Más insatisfecho que satisfecho.

5 Máxima insatisfacción.

6 Contradictoria.

Tabla 19. *Cuadro lógico de Iadov. (Fuente: Elaboración propia)*

¿Le gustaría hacer uso de la herramienta MelCP, para evaluar los proyectos de curso y los estudiantes que lo integran?	**¿Considera usted oportuno continuar efectuando las evaluaciones de los trabajos de curso y sus estudiantes manualmente, a pesar del gran volumen de información que se genera?**								
	No			No sé			Sí		
	¿La herramienta MelCP contribuye a la interpretabilidad de las evaluaciones de los estudiantes acorde a sus necesidades?								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta mucho	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	6
No sé qué decir	2	3	6	3	3	3	6	3	4

Para obtener el índice de satisfacción grupal (ISG) se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y - 1 de la siguiente forma:

Tabla 20.*Resultado de aplicación de la técnica Iadov. (Fuente: Elaboración propia)*

Índice de satisfacción	**Escala**
Máxima satisfacción	+1
Más satisfecho que insatisfecho	0,5
No definido y contradictorio	0
Más insatisfecho que satisfecho	-0,5
Máxima insatisfacción	-1

La satisfacción grupal (ISG) se calcula por la siguiente fórmula:

$$ISG = \frac{A(+1) + B(+0{,}5) + C(0) + D(-0{,}5) + E(-1)}{N}$$

Donde:

A representa el número de sujetos con índice individual 1

B representa el número de sujetos con índice individual 2

C representa el número de sujetos con índice individual 3 ó 6

D representa el número de sujetos con índice individual 4

E representa el número de sujetos con índice individual 5

N representa el número total de sujetos del grupo

Como resultado se obtuvo:

$ISG = \frac{6*1+1*0,5}{4}$

De manera que el ISG = 0,93

Los resultados de la satisfacción individual según las categorías empleadas fueron los siguientes:

***Tabla 21**.Resultado de aplicación de la técnica Iadov. (Fuente: Elaboración propia)*

Nivel de satisfacción	Cantidad	%
Máxima satisfacción	6	93,%
Más satisfecho que insatisfecho	1	7%
No definida	0	0

Al procesar las respuestas a las encuestas en el cuadro lógico de Iadov, se obtiene un grado de satisfacción grupal de 0.93, lo cual se traduce en una clara satisfacción con el uso de la herramienta informática MELCP.

En el criterio respecto a la interpretabilidad en el MELCP a través del uso de la solución propuesta, hubo una concordancia de un 100% en cuanto a que contribuye a su mejora. De igual manera el 100% de los participantes manifestó que le gustaría mucho hacer uso del MELCP para desarrollar los procesos de evaluación. Mientras que un 7% no sabe si sería oportuno continuar ejecutando los procesos de evaluación mediante la herramienta MELCP.

Las preguntas abiertas que se formularon fueron:

- ¿Qué valoraciones le sugiere a la herramienta MELCP respecto a la interpretabilidad de los resultados?
- ¿Qué elemento(s) usted adicionaría a la solución que se propone?

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

- El módulo permite un control más eficiente de los procesos asociados a la evaluación de los trabajos de curso.
- El módulo les garantiza a los profesores rapidez para acceder a la información.
- EL módulo permite realizar un seguimiento constante de las evaluaciones.

La aplicación de la técnica de Iadov aportó información significativa respecto al grado de satisfacción del cliente. Los resultados obtenidos y los criterios emitidos validan la fortaleza de la propuesta, reflejándose una valoración muy positiva del cliente con la solución.

Luego de los resultados obtenidos en la validación de las variables y la aplicación de la técnica Iadov, el cliente emite su acta de aceptación, lo que evidencia su satisfacción con la herramienta MELCP. Para ver detalles del acta de aceptación de la herramienta MELCP Anexo 2.

3.4 Conclusiones parciales:

En este capítulo se diseñó un caso de estudio con el propósito de validar la propuesta de solución presentada en el capítulo anterior. Tras aplicar las pruebas de software definidas y el diseño experimental descrito, se concluye lo siguiente:

- La confección de las tareas de ingeniería por iteración y de las tareas detalladas permitieron simplificar la implementación de las historias de usuario.
- El uso de estándares de codificación permitió proyectar calidad en el código generado.
- Las pruebas de caja blanca realizadas a varios algoritmos del sistema posibilitaron la detección de errores, obteniéndose resultados satisfactorios.
- Las pruebas de caja negra permitieron comprobar el correcto funcionamiento de la herramienta luego de realizadas dos iteraciones.
- La validación de la solución demostró que la herramienta desarrollada contribuye al tratamiento de la incertidumbre en la evaluación del aprendizaje en los trabajos de curso.

CONCLUSIONES

Finalizada la presente investigación se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, resaltando que:

- El uso de las técnicas de soft computing para la evaluación de proyectos de curso mejora la posición de los profesores a la hora de emitir una evaluación.
- El análisis del experimento aplicado demostró que la aplicación facilita a los profesores el proceso de evaluación de trabajos de curso, disminuyendo el tiempo del mismo, evitando la pérdida de información y facilitando la interpretabilidad de los resultados obtenidos.
- La herramienta informática desarrollada contribuye a facilitar a los profesores la evaluación de trabajos de curso basada en técnicas de soft computing que contribuye a mejorar la interpretabilidad de los resultados para emitir la evaluación del estudiante.

RECOMENDACIONES

- Integrar la solución con los servicios de la UCI de manera tal que se puedan importar los estudiantes y profesores que se encuentran en sus servidores.
- Implementar una funcionalidad que permita al profesor principal evaluar a los evaluadores existentes en el sistema.
- Implementar una funcionalidad que permita a los estudiantes acceder al sistema y poder observar las notas emitidas por cada uno de sus profesores.

REFERENCIAS BIBLIOGRÁFICAS

Almeira, Adriana Sandra y Pérez, V. 2007. *Arquitectura de Software: Estilos y Patrones.* Argentina : s.n., 2007.

Arturo Guerra, César. 2018. CG. *Obtención de Requerimientos. Técnicas y Estrategia.* [En línea] 18 de Febrero de 2018. https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

Christiansson, Benneth. 2008. *GoF Design Patterns-with examples using Java and UML2.* 2008.

Cuervo Álvarez, José. 2015. Informática jurídica. [En línea] 1 de enero de 2015. http://www.informatica-juridica.com.

Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A. 2010. *Metodologías tradicionales vs. metodologías ágiles.* 2010.

FireOS. 2017. Servidor Web Apache: ¿Qué es, cuáles son sus caracteristicas y para qué se utiliza? [En línea] Ibrugor, 8 de marzo de 2017. http://www.ibrugor.com/blog/apache-http-server- que-es-como-funciona-y-para-que-sirve/..

García Vergara, Alvaro. 2014. *La utilización de la herramienta Evalcomix en la Evaluación de Competencias de los Trabajos Fin de Grado y Máster.* 2014.

García, Ramos. 1986. Bases pedagógicas de la evaluación. Madrid : s.n., 1986. s.n.

Google Docs. 2015. Estándares de Codificaciónv1[1].3.doc. [En línea] 2015. https://docs.google.com/document/d/.../edit?hl=en_US.

Herramienta para modelado UML. 2016. Paraiso Linux. *Herramienta para modelado UML.* [En línea] 2016.

Herrera, Francisco and Martínez, Luis. 2000. A 2-tuple fuzzy linguistic representation model for computing with words. 2000. Vol. 8, 6, pp. 746-752.

IDA. 2016. {ida BLOG. [En línea] 7 de enero de 2016. http://ida.cl/blog.

Joskowicz, Jose. 2012. *Reglas y prácticas en eXtreme Programing.* 2012.

Juan Pablo. 2013. el Conspirador. *Un blog de volcanes, exploraciones y tecnología.* [En línea] 21 de 12 de 2013. https://www.elconspirador.com/2013/12/21/que-es-y-para-que-sirve-un-modelo-conceptual/.

Jummp. 2012. Jummp. *Gestión de proyectos y desarrollo de software.* [En línea] 10 de enero de 2012. https://jummp.wordpress.com/2012/01/10/desarrollo-de-software-tarjetas-crc/.

Kidd, Lorenz Mark y Jeff. 1994. *Object-Oriented Software Metrics.* New Jersey : s.n., 1994.

Larman, Craig. 1999. *ML y Patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : Pablo Eduardo Roig Vázquez, 1999. ISBN: 970-17-0261-1.

Leyva Barajas, Dra. Yolanda Edith. 2010. Evaluación del Aprendizaje: Una guía práctica para profesores. 2010.

Lucid. 2018. Lucidchat. [En línea] 2018. ISBN: 84-7829-037-0..

Managing non-homogeneous information in group decision making. **Herrera, F., Martínez, L. and Sánchez, P.J. 2005.** 166, 2005, European Journal of Operational Research, pp. 115-132. ISSN: 0377-2217.

Martínez, Luis y Herrera, Francisco. 2012. An overview on the 2-tuple linguistic model for computing with words in decision making: Extensions, applications and challenges. 2012, Vol. 207, 1, págs. 1-18.

Martínez, Martínez y Ruan, L. y y Herrera F., D. 2010. Computing with Words in Decision support Systems: An overview on Models and Applications. 2010, Vol. 3, s.n.

Mateu, Carles. 2004. *Desarrollo de aplicaciones web.* 2004.

REFERENCIAS BIBLIOGRÁFICAS

MES, Ministerio de Educación Superior. 2007. Resolución No. 210/07. *Reglamento para el Trabajo Docente y Metodológico.* La Habana, La Habana , Cuba : Gaceta Oficil, 08 de Agosto de 2007.

Metodologías ágiles de gestión de proyectos . 2008. Metodologías ágiles de gestión de proyectos (Scrum, DSDM, Extreme Programming – XP…). 2008.

N. V., Kuzmina. 1970. *Metódicas Investigativas de la actividad pedagógica.* Leningrado : s.n., 1970.

NetBeans. 2015. NetBeans. [En línea] 2015. https://netbeans.org/index_es.html..

Peña Abreu, Dra. C. Marieta y Rodríguez Rodríguez, MSc. Carlos Rafael. 2018. *Método para la evaluación del aprendizaje en los proyectos de curso aplicando computación con palabras.* 2018.

Pérez García, Alejandro Alfonso. 2007. *Desarrollo de herramienta web de gestión docente.* 2007.

PÉREZ GÓMEZ, Ángel I. 2010. *Aprender a educar. Nuevos desafíos.* Málaga : Revista Interuniversitaria de Formación del Profesorado, 2010. ISSN 0213-8646.

PHP. 2015. PHP.net. [En línea] 2015. http://www.php.net/manual/es/intro-whatis.php..

Pressman, Roger S. 2010. *Ingeniería del Software. Un Enfoque Práctico.* 2010.

Recaman, Hernando Chaux. 2012. *Marco de trabajo para aplicaciones web de código abierto en instituciones universitarias.* 2012.

Rodriguez, Tamara. 2014. *Metodología de desarrollo para la Actividad productiva de la UCI.* La habana : s.n., 2014.

Sommerville, Ian. 2011. *Software Engineering.* 2011. ISBN-13: 978-0-13-703515-1, ISBN-10: 0-13-703515-2.

Symfony. 2014 . Symfony . *Symfony 3.0: The roadmap.* [En línea] 11 de Noviembre de 2014 . http://symfony.com/blog/symfony-3-0-the-roadmap.

TechTarget. 2016. SearchDataCenter. [En línea] mayo de 2016. https://searchdatacenter.techtarget.com/es/definicion/Modelado-de-datos.

TicherVirtual. 2016. TicherVirtual. *Quizalize: ua herramienta divertida para evaluar.* [En línea] 14 de junio de 2016. http://tichervirtual.com.

Tió Torriente, Lázaro, y otros. 2011. *Instrumento y herramienta informática para guiar, controlar y evaluar las interacciones de los estudiantes en foros virtuales.* 2011. ISSN 0864-2141.

UNAD. 2013. Universidad Nacional Abierta y a Distancia. [En línea] 2013. http://stadium.unad.edu.co/ovas/10596_9836/diagramas_de_componentes.html.

Verdegay, J. L. 2005. De los conjuntos fuzzy a la soft computing. Santiago de Compostela : s.n., 2005. Vol. 24.

Zadeh, L. A. 1994. *Soft computing and fuzzy logic. Software.* 1994. IEEE.

—. 1998. *Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems. Soft Computing.* 1998.

Zadeh, Lotfi A. 1996. NACIMIENTO Y EVOLUCIÓN DE LA LÓGICA BORROSA, EL SOFT COMPUTING Y COMPUTACIÓN CON PALABRAS: UN PUNTO DE VISTA PERSONAL. *Psicothema.* 1996, Vol. 8, 2.