INSTITUTO SUPERIOR MINERO METALURGICO

“Dr. Antonio Núñez Jiménez”.

Facultad de Metalurgia - Electromecánica

Moa, Holguín

HERRAMIENTA DE CONTROL Y BENEFICIOS PARA PROYECTOS.

Trabajo de diploma para optar por el título de Ingeniería en Informática

Autor (es): **Yoangel Hernández García.**

Tutor (es): **Lic. Delmis Wilfredo Carbonell Laborde.**

MSc. Berto Andrés Martín Téllez.

Consultante (es):

Moa, Cuba

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al *Instituto Superior Minero Metalúrgico de Moa "Dr. Antonio Núñez Jiménez" (ISMMM) y a DESOFT S.A. División Guantánamo* a que den el uso que estimen pertinente a este trabajo.

Para que así conste firmo la presente a los _____ días del mes de _____ del 2010.

Yoangel Hernández García.
_____
Nombre completo del autor

Delmis Wilfredo Carbonell Laborde.
_____
Nombre completo del primer tutor

Berto Andrés Martín Téllez.
_____
Nombre completo del segundo tutor

Agradecimientos:

A todas mis amistades, compañeros de aula y profesores que han estado a mi lado en estos 5 años, que confiaron en mí y me apoyaron en el desarrollo de esta investigación.

A toda mi familia que nunca dudó que este momento llegaría a mi vida.

A todos los compañeros de la empresa Desoft división Guantánamo, en especial a un viejo peleón pero muy buena gente: Luís H. Batista, siempre me dio su mano cuando la necesité; a Delmis W. Carbonel, mi tutor y a Onésimo Mustelier; a los demás en el departamento de Desarrollo, para que no se me pongan celosos.

A mi única hermana, Yaritza, tan diferente de mí, a la que quiero mucho.

A mi novia, que aunque no lo crean debería ponerla como autora de este documento de tantas veces que me inspiró realizarlo.

A mis abuelos, aunque la mayoría de ellos no pueda estar conmigo en este momento, gracias por traer a este mundo y poner en el mismo camino a unos padres como los que tengo.

El agradecimiento especial e interminable a mis padres: Olga y Papo, que han sido, son y serán la luz que me guía.

En especial a una niñita que no sabe leer aun, pero fue mi inspiración en los momentos de tensión en estos 5 años de universidad, mi sobrina Erika.

A todos muchas gracias.

Dedicatoria

Dedico esta tesis a dos personas que forman parte de mí desde mi misma creación, uno formó mi lado izquierdo, el otro el derecho, las dos mitades de mi corazón; mis padres José Ángel y Olga, ellos serán los verdaderos propietarios de la felicidad que pueda traer esta tesis.

Pensamiento

- *La mayoría de las ideas fundamentales de la ciencia son esencialmente sencillas y, por regla general pueden ser expresadas en un lenguaje comprensible para todos.*

- *No busques ser alguien de éxito sino busca ser alguien valioso: lo demás llegará naturalmente.*

- *Intenta no volverte un hombre de éxito, sino volverte un hombre de valores.*

- *El amor por la fuerza nada vale, la fuerza sin amor es energía gastada en vano.*

Albert Einstein

Resumen

Debido a la ineficiente planificación y control para realizar un software y la existencia de una metodología poco satisfactoria en la empresa Desoft S.A., se hace indispensable la realización de una herramienta para el control y beneficios en proyectos realizados por la división Guantánamo mediante la utilización de la metodología de Marco Lógico.

Dicha metodología ha sido probada satisfactoriamente en varias empresas ya sean del mundo capitalista como del socialista, probando ser eficiente.

Este trabajo de tesis tiene como objetivo alcanzar el perfeccionamiento en Desoft logrando un pensamiento avanzado de los trabajos que se realizarán y la disminución del tiempo a la hora de la planificación de proyectos a desarrollar por la empresa.

Abstract

Due to inefficient planning and control for software making and the existence of an unsatisfactory methodology in the DESOFT SA Enterprise, it is indispensable the creation of a tool for the control and benefits of projects undertaken by the Guantánamo branch by the use of Logical Framework methodology.

This methodology has been successfully tested in several companies in the world whether capitalist and socialist, proving to be efficient.

This thesis aims to achieve improvement in DESOFT with a forward thinking of works to be performed and the reduction of time when planning projects to be developed by the Enterprise.

Índice

Tablas

***Introducción*.**

En este estudio se abordan aspectos básicos relacionados con el tema del control y gestión de la información en una empresa con respecto a proyectos en desarrollo, centrados en la utilización de la metodología de **Marco Lógico**. Se analizan conceptos, características, ventajas y beneficios de su utilización; estos elementos constituyen el soporte teórico del proyecto, y permiten un mejor entendimiento de la situación problémica y de una valoración adecuada para su solución.

La metodología de marco lógico es la más completa y utilizada en gran parte del mundo en la actualidad para conceptuar, diseñar, ejecutar, seguir el desempeño, evaluar y comunicar información fundamental sobre proyectos de forma resumida y exhaustiva ya que da elementos para estructurar el proceso de planificación de cualquier proyecto a desarrollar. Se suma a esto que las principales organizaciones para el desarrollo, como el Ministerio de Economía y Finanzas así como otras entidades financieras globales requieren que se presenten los proyectos bajo la forma de un Marco Lógico.

La experiencia internacional de los últimos 40 años es prueba fehaciente de la validez del enfoque en proyectos para la promoción del desarrollo y de su utilidad en la gestión del ciclo de los proyectos, particularmente en su diseño. Esta metodología propone un método para organizar y visualizar la interacción de los distintos elementos de un proyecto. Para este enfoque, los recursos humanos y materiales, expresados ambos en términos físicos o monetarios, constituyen los insumos básicos para que funcionen las actividades, que permiten a su vez obtener ciertos productos. Estos elementos constituyen, en rigor, el proyecto y están bajo control y responsabilidad de la institución ejecutora. Los productos obtenidos (también llamados componentes, resultados u objetivos instrumentales del proyecto), tienen un efecto predecible bajo ciertas condiciones del entorno, sobre los beneficiarios directos, lo cual es descrito en el propósito u objetivo inmediato y más ampliamente, en el fin u objetivo global del proyecto.

La automatización de la metodología de Marco Lógico permitirá elevar la calidad de los proyectos de desarrollo en la empresa Desoft S.A. División Guantánamo, lo cual esta dado por los niveles de precisión en el orden metodológico, organizativo y para el control de la calidad que ofrece el Marco Lógico, ello permitiría a Desoft S.A mayores oportunidades competitivas en el mercado internacional, así como la aplicación de la ciencias en función de un resultado productivo concreto.

Situación Problémica: No existencia de un software en la empresa Desoft S.A. que facilite la organización, planificación y análisis de proyectos de servicios informáticos, a partir del empleo de la metodología del Marco Lógico.

Debido a esta situación problémica surge el siguiente **Problema científico**: La carencia de un software en la empresa DESOFT S.A. División Guantánamo para la organización, planificación, análisis y control de proyectos de desarrollo afecta la calidad de los servicios informáticos en dicha empresa.

Este problema se enmarca en el **objeto de estudio**: Software que permita la organización, planificación, análisis y control de proyectos a partir de la metodología del Marco Lógico.

Para dar solución al problema planteado se propone como **Objetivo general:** Elaborar un software que permita la organización, planificación, análisis y control de proyectos de desarrollo de software para elevar la calidad de este servicio en la empresa Desoft División Guantánamo.

Campo de acción: Los proyectos de desarrollo de software en la empresa Desoft División Guantánamo.

Para guiar esta investigación se plantea la siguiente **Hipótesis**: Con la existencia de un software a partir de la metodología Marco Lógico para la organización, planificación, análisis y control de proyectos de desarrollo de software en la empresa

Desoft División Guantánamo se podrá elevar la calidad de los servicios informáticos que aquí se brindan.

De acuerdo a esta propuesta de derivan los siguientes **Objetivos específicos.**

- ✓ Realizar el análisis de la herramienta propuesta.
- ✓ Diseñar la herramienta propuesta.
- ✓ Implementar el Sistema.
- ✓ Validar la solución propuesta.

Para el logro de los objetivos fue necesario plantearse las siguientes **Tareas:**

- ✓ Estudio preliminar y búsqueda de información.
- ✓ Diseño de la base de datos.
- ✓ Diseño de la arquitectura del sistema.
- ✓ Diseño de los diagramas acorde a la metodología propuesta.
- ✓ Programación de la interfaz.
- ✓ Análisis de Factibilidad y Sostenibilidad.

La metodología utilizada es fundamentalmente cualitativa, se emplearon métodos empíricos y teóricos.

Métodos empíricos:

Entre los métodos empíricos usados podemos citar **la entrevista y la observación** para la recopilación de la información. La entrevista permitió determinar los principales requerimientos del sistema y funcionalidades que necesitan quedar plasmadas en las historias de usuarios. La observación fue útil para entender el comportamiento del sistema y sus especificaciones.

Métodos teóricos:

Entre los métodos teóricos podemos encontrar:

- ✓ **Análisis y síntesis:** mediante el análisis y síntesis de la documentación disponible conocimos el funcionamiento actual de la matriz de gauss y la confección del informe final.
- ✓ **Hipotético-deductivo:** en la elaboración de la hipótesis, a partir de la cual se realizaran deducciones que arriben a la solución del problema.
- ✓ **Histórico-lógico:** para investigar el desarrollo que ha tenido el tema (antecedentes) y apoyar los conocimientos que sobre este existen.
- ✓ **Modelación:** la modelación permitió realizar una representación de la realidad, se logró detectar problemas en la forma actual de procesar la información y encontrar las funcionalidades que debe tener el sistema que se propone, que lo harán más completo y le brindarán satisfacción al usuario con un producto de mayor calidad.

El sistema Marco Lógico permite realizar un diseño que satisface tres requerimientos fundamentales de calidad de un proyecto de desarrollo: **coherencia, viabilidad y evaluabilidad**. Su creciente popularidad se debe al importante hecho de constituir la principal técnica no cuantitativa de análisis científico en el campo de la política del desarrollo. En la actualidad el enfoque del Marco Lógico es utilizado por la mayoría de las agencias de cooperación y desarrollo a nivel mundial, tales como el Banco Interamericano de Desarrollo (BID) y el Banco Mundial (BM), por organismos internacionales (ONU), agencias de cooperación (AUSAID, CIDA) y países (Chile, Colombia y los de la Unión Europea).

Como parte del objeto de estudio, se expone la situación actual de la gestión de los negocios asociados con proyectos o programas y de la forma en que se lleva a cabo actualmente el proceso de desarrollo en la empresa productora de software (Desoft S.A División Guantánamo); se presentarán además las características generales de los sistemas vinculados al campo de acción.

Capítulo 1

FUNDAMENTACIÓN DEL TEMA.

1.1 Introducción.

En las empresas del país, para el desarrollo de cualquier proyecto propuesto, la gestión para el diseño de proyectos y programas no se realiza de forma automatizada, ni con una metodología concreta como podría ser la metodología Marco Lógico, la que por su eficiencia probada, en el ámbito internacional, brinda elementos indispensables para alcanzar niveles más elevados y eficientes en el resultado final de un proyecto. Al aplicar esta metodología en nuestro país se lograría en consecuencia un alto desarrollo y perfeccionamiento en la planificación de softwares. La implementación de este sistema permitirá el control y la gestión de los proyectos definidos en la empresa Desoft División Guantánamo. A través de este sistema se podrá controlar la utilidad de antemano en torno a diversos marcos que serán definidos por el individuo que interactúe durante la planificación de un proyecto.

La metodología Marco Lógico surge en 1969, bajo la firma consultora Practical Concepts Inc., específicamente por León Rossenberg y Lawrence Posner, contratados por la Agencia Internacional de Desarrollo de los Estados Unidos (USAID), con el fin de mejorar la calidad de las inversiones sociales, superando los tres problemas que en opinión de sus creadores eran los principales defectos de los proyectos de desarrollo:

- Planificación demasiado imprecisa.
- Responsabilidad gerencial ambigua.
- Evaluación excesivamente controversial.

A finales de 1997 esta metodología fue rediseñada por la Agencia Alemana de Cooperación Técnica (**GTZ)**, bajo el nombre de Planificación de Proyectos Orientada a Objetivos (ZOPP), la cual incorporó nuevos elementos a la concepción original del

Marco Lógico, como el análisis de participantes, análisis de problemas, análisis de objetivos y análisis de alternativas. El trabajo en equipos multidisciplinarios mediante talleres en los que tomaban parte la GTZ, las organizaciones contrapartes y los grupos beneficiarios también fue incorporado en lo que constituyó una metodología participativa de diseño de proyectos.[1]

1.2 Estado del Arte.

1.2.1 Conceptos Fundamentales

1.2.1.1 ¿Por qué diseñamos proyectos y programas?[2]

- Los proyectos se diseñan porque existe un problema de desarrollo, un obstáculo al desarrollo.
- Ese obstáculo se genera porque existe un servicio público deficiente o porque no existe dicho servicio. Muchas veces hay consenso de que la situación vigente es insatisfactoria, que se requiere un cambio. A esa situación insatisfactoria la llamamos **Situación Actual**.
- Si existe una situación actual insatisfactoria, podemos decir que hay también una **Situación Futura Deseada,** esta sería el resultado de una **intervención** diseñada para mejorar algunos o todos los elementos de la situación actual.
- Esa intervención es un **proyecto** o un **programa**, que se ejecuta en el corto y mediano plazo para lograr en el mediano y largo plazo la situación deseada.

[1] Citado de:
- Curso Breve De Marco Lógico.(Presentación de Power Point)
- http://www.fondoempleo.com.pe/marcologico.htm 20 febrero 2010.

[2] Citado de:
http://es.wikipedia.org/w/index.php?title=Marco_L%C3%B3gico&redirect=no 21 febrero 2010.

1.2.1.2 Las dos herramientas para diagnosticar de la manera más objetiva posible la Situación Actual son:[3]

- El análisis de involucrados.
- El análisis de problemas.

Mediante estos dos pasos se alcanza la identificación del problema.

1.2.1.3 Las dos herramientas para especificar la Situación Deseada son:

- El análisis de objetivos.
- El análisis de alternativas.

El resultado de estos pasos es la identificación de un proyecto.

1.2.1.4 Marco Lógico

El Sistema de Marco Lógico es una de las principales metodologías utilizadas por las instituciones para diseñar y planificar sus proyectos o programas, se compone de una secuencia de 5 pasos metodológicos.

1.2.1.5 Los pasos metodológicos del Marco Lógico son:

1. El Análisis de Involucrados.
2. El Análisis de Problemas.
3. El Análisis de Objetivos.
4. El Análisis de Alternativas.
5. La Matriz del Marco Lógico.[4]

1.2.1.5.1 Análisis de Involucrados.

Este análisis se hace para identificar y esclarecer qué grupos y organizaciones están involucrados directa o indirectamente en el problema de desarrollo específico que

[3] Citado de: Curso Breve De Marco Lógico.

[4] Citado de:
- Curso breve de Marco Lógico.
- http://liceoindustrial2008.wordpress.com/proyecto-de-marco-logico/ 21 febrero 2010.
- http://www.monografias.com/trabajos27/marco-logico/marco-logico.shtml 21 febrero 2010.
- http://www.slideshare.net/guestaaf1b8/el-marco-logico 22 febrero 2010.

intentamos resolver, para tomar en consideración sus intereses, su potencial y sus limitaciones.

1.2.1.5.2 El Análisis de Problemas.

- Analizar la situación actual relacionada con el problema de desarrollo seleccionado.
- Identificar los problemas principales en torno al problema de desarrollo y las relaciones causa-efecto entre ellos.
- Visualizar las relaciones de causalidad y sus interrelaciones en un diagrama (árbol de problemas).

1.2.1.5.3 El Análisis de Objetivos

- Describir una situación que podría existir después de resolver los problemas.
- Identificar las relaciones medio-fin entre objetivos.
- Visualizarlo en un diagrama.

1.2.1.5.4 El Análisis de Alternativas

El análisis de alternativas consiste en identificar estrategias alternativas a partir del árbol de objetivos que, si son ejecutadas, podrían promover el cambio de la situación actual a la situación deseada.

Después de identificadas las distintas estrategias se debe evaluar cada una con varias herramientas de análisis que en realidad son filtros para ir seleccionando las alternativas deseadas.

1.2.1.5.5 La Matriz de Marco Lógico

- Resume todo lo discutido en los cuatro pasos anteriores.
- Se agrega información sobre lo que se va a monitorear.
- Se agrega información sobre lo que se va a evaluar.
- Da a conocer el alcance de la responsabilidad del gerente del proyecto.

1.2.1.5.6 La Matriz de Marco Lógico es una herramienta para:

- La concepción.
- El diseño.
- La ejecución.
- El seguimiento de desempeño.
- La evaluación de un proyecto.[5]

1.2.2 Antecedentes

En investigaciones efectuadas por los distintos profesionales e investigadores de la empresa se detectó que no existen antecedentes de una herramienta informática a partir del Marco Lógico que permita el control y beneficios para proyectos y en particular los de desarrollo de software, así como que controle el comportamiento presente y futuro de cualquier proyecto planificado con el objetivo de controlar la ejecución y el comportamiento mediante la metodología planteada, sólo hay constancia de elementos o formas de realización y algunas guías para la confección de proyectos acerca de esta metodología para su puesta en práctica[6], por lo que la herramienta que propone este trabajo asegura una posible colocación en la cartelera de productos de la Empresa DESOFT S.A. para la posterior venta a empresas nacionales por parte de la División Guantánamo y su empleo en los proyectos actuales y futuros de exportación de software.

1.3 Sistemas automatizados existentes vinculados al campo de acción

Como se ha visto anteriormente no se han encontrado antecedentes que indiquen la existencia de un software relacionado con el objeto de automatizar el proceso de planificación y control a proyectos y en particular los de desarrollo de software. No obstante los sistemas más utilizados para dar seguimiento a cualquier proyecto de

---

[5] Citado de Algunas Ideas Claves para la gestión de proyectos internacionales de las universidades cubanas, Carlos Alberto Vigil Taquechel.

Otras citas utilizadas: ´

- Logical Framework (Logframe) Methodology, de Herman H. Grant.
- Matriz de indicadores y gestión de programas, de Eduardo Aldunate.

J.J Gutierrez, M.J Escalona, M. Mejias, J.Torres. Pruebas del sistema en programación extrema.

[6] Un ejemplo de estas guías es: GUÍA PARA LA CONFECCION DE LOS PROYECTOS

forma resumida son los que en su lógica contengan la metodología planteada, ofreciendo elementos para estructurar el proceso de planificación y las principales organizaciones para el desarrollo.

1.4 Tendencias y tecnologías actuales

1.4.1 Metodología de desarrollo vinculada con los aspectos a tratar en este proyecto.

La metodología **XP**, llamada así por sus siglas en inglés (Extreme Programming) es una metodología ágil de desarrollo de software, orientada a la gente que produce y usa el software. Reduce el costo del cambio en las etapas de vida del sistema. Combina las que han demostrado ser las mejores prácticas de desarrollo de software, y las lleva al extremo. XP fue creado por Kent Beck para la plantilla del proyecto C3 en Chrysler, Kent fue contratado para dirigir el proyecto, durante el proceso nació una nueva metodología: Extreme Programming (XP), C3 concluyó exitosamente en 1997. XP es una colección de reglas y prácticas mutuamente soportadas. Usadas en conjunto, definen una metodología.

El proceso de desarrollo SW se divide en cuatro tipos de actividades:

- ***Planificación.***

  - Se escriben los relatos de usuario ("user stories").
  - El plan de entregas crea el cronograma.
  - Entregas pequeñas muy frecuentes.
  - Se mide la velocidad del proyecto.
  - Se divide el proyecto en fases de iteración.
  - La planificación de iteración inicia cada fase.
  - El personal rota por las diferentes áreas del sistema.
  - Reuniones de pie, todos los días.
  - Corregir el XP si falla (cambiar las reglas por consenso).

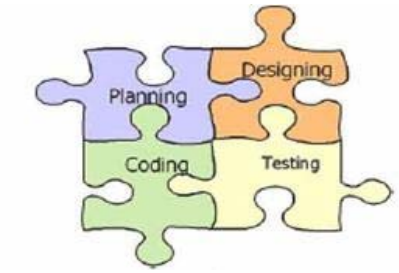
- ***Diseño.***

- Simplicidad.
- Elegir una "metáfora" de sistema, un conjunto de nombres ilustrativos de la realidad.
- Usar tarjetas CRC en las sesiones de diseño. [7]
- Crear soluciones rápidas puntuales ("spike solutions") para reducir el riesgo.
- No agregar funcionalidad antes de tiempo.
- Refactorizar donde y cuando sea posible.
- Codificación.
- El cliente siempre al alcance.
- El código debe respetar las normas aceptadas.
- Codificar primero las pruebas de unidad (unit tests).
- Todo el código se escribe en parejas de programadores.
- Integra código sólo una pareja por vez.
- Integrar seguido.
- El código es propiedad colectiva.
- No optimizar hasta el final del proyecto.
- No trabajar horas extra.

❖ ***Prueba.***

- Todo el código debe tener pruebas de unidad.
- Todo el código debe pasar las pruebas de unidad antes de la entrega.
- Crear pruebas toda vez que aparezcan errores.
- Correr frecuentes pruebas de aceptación y publicar la puntuación.

[7] Lo anteriormente dicho sobre la metodología de desarrollo XP ha sido tomado y citado de:
- Introducción a la programación extrema, Msc. Alejandro Aguilera Sierra.
- Programación extrema, Wilder Escobedo D.



XP en detalle.

Relatos de usuario.

a) Similares a los Casos de Uso pero no es lo mismo; estiman el tiempo con vista al plan de entrega.
b) Las escriben los clientes, describen algo que el sistema debe hacer, son 2 o 3 oraciones en el lenguaje del usuario, sin jerga técnica, con poco detalle. La descripción detallada se dará al momento de iniciar el desarrollo.
c) Definen una o más pruebas de aceptación.
d) Corresponden a 1, 2 ó 3 semanas de desarrollo a tiempo completo.
e) En un plan de entrega típico, se definen unas 80 historias ± 20%.

Planificación de entrega.

a) En una reunión de planificación se define un plan de entrega para todo el proyecto. El plan de entrega se usará para crear plantes de iteración.
b) Se toman decisiones de desarrollo y de negocios, se negocia un cronograma factible por todos.
c) Se parte de las relatos de usuario, impresos o en tarjetas; moviéndolas en la mesa se define el primer (o el siguiente) conjunto a implementar, algo utilizable, verificable, con sentido para la empresa, que se pueda entrgar a corto plazo.
d) Puede planearse por tiempo (cuántas historias antes de tal fecha) o por alcance (cuánto tiempo para estas historias), a la velocidad de programación estimada para el equipo.

e) Las iteraciones se planifican en detalle recién al encararse, no antes. Las reglas están en el Portland Pattern Repository.
f) Negociar hasta la coincidencia de clientes, desarrolladores y gerentes; no cambiar las estimaciones de los relatos de usuario.
g) Un proyecto puede ser medido en 4 variables: 1) Alcance: qué se hará; 2) Recursos: gente disponible; 3) Fecha de entrega; 4) Calidad: bien hecho, bien probado. Sólo pueden elegirse 3 de las 4; la 4a. resulta de las anteriores. Además, la calidad no debiera cambiarse.

El Plan de Entrega.

a) Define los conjuntos de relatos de usuario a implementar en cada iteración, y sus fechas de entrega.
b) Cuando se verifica un cambio de la velocidad de programación durante 2 ó 3 iteraciones deberá rehacerse el Plan de Entrega.

Entregas frecuentes.

a) El Plan de Entrega busca definir unidades funcionales con sentido para producir entregas frecuentes.

Velocidad de programación.

a) Es la cantidad de relatos de usuario o tareas de programación realizadas en una iteración (un simple conteo).
b) Es inútil sacar medias por programador o por tarea; cada equipo tiene sus propias características.
c) Se esperan variaciones en la velocidad. Si son extremas, rehacer el Plan de Entrega.

Desarrollo de una iteración.

a) Dividir el Plan de Entrega en unas 12 iteraciones, más o menos. Cada iteración demora de 1 a 3 semanas.
b) No adelantar planificación: cada iteración se planifica inmediatamente antes de comenzar. No adelantar implementación, desarrollar sólo lo previsto en la presente iteración.

Plan de iteración.

a) Incluye los relatos de usuario y las pruebas de aceptación. Ambas se dividen en tareas de programación. Una tarea de programación abarca de 1 a 3 días de trabajo ideales, sin interrupciones ni otras actividades.
b) El programador estima la tarea que toma.
c) Estimar el tiempo de acuerdo a la velocidad de programación ya verificada. Agregar o quitar tareas si no alcanza el tiempo; si es preciso se renegociará. No engañarse ignorando o alterando el cálculo frío.
d) Insistir en implementar primero lo esencial; si no se llega será más fácil renegociar los accesorios.
e) No sacrificar la refactorización ni las pruebas de unidad: descuidar esto atrasará más.

Rotación del personal.

a) Mover al personal de un área a otra para evitar pérdidas de conocimiento o cuellos de botella en la codificación; todos deben conocer bastante código de otras secciones. La rotación y la programación en parejas produce el entrenamiento cruzado que evita las "islas de conocimiento".
b) En cada iteración todos deben trabajar parte del tiempo en algún área nueva. En cada pareja de programadores se rota sólo uno por vez.

Reunión diaria de pie.

a) Organizar una única reunión diaria, breve, de pie, con todos los integrantes, sólo para plantear problemas. Estos se resolverán luego en pequeños grupos de involucrados.

Corregir XP.

a) Adaptar las reglas cuando no se obtienen los resultados esperados. Las reglas se pueden cambiar pero no pueden dejar de existir: cada desarrollador debe saber exactamente qué puede esperar de los demás, y los demás de él.

Buscar la simplicidad.

a) Un diseño simple es más fácil de crear y mantener. Buscar afanosamente el diseño más simple que funcione. No implementar funcionalidad antes de tiempo.
b) Advertencia: lograr un diseño simple es un gran trabajo.

Esquema de nombres ("system metaphor").

a) Nombrar clases y métodos en forma consistente y clara. Debe ser posible inferir la significación real de un nombre nunca visto antes, a qué cosa se refiere.

Tarjetas CRC.

a) Usar tarjetas CRC, "Clase, Responsabilidad, Colaboración", en el diseño grupal. Ayudan a evitar el enfoque procedimental y destacan la orientación a objetos.
b) Cada tarjeta CRC representa un objeto. El nombre de Clase va arriba, las responsabilidades (qué debe hacer) a la izquierda, las clases asistentes (que colaboran) a la derecha.
c) No suele ser necesario escribir la tarjeta completa; los participantes se familiarizan rápidamente con el propósito de cada clase.

d) En la reunión CRC alguien simula el sistema discutiendo los mensajes intercambiados entre objetos. Limitar a 1 o 2 personas de pie exponiendo, mientras los demás permanecen sentados.

Solución rápida ("spike solution").

a) Crear una solución rápida, un programa muy simple para explorar una solución posible, enfocada en el problema central sin atender el resto. Suele ser descartado; se emplea para probar la solución, mejorar la estimación, reducir el riesgo potencial.

Nunca agregar funcionalidad extra.

a) No agregar funcionalidades extra sólo porque ahora se ve claro como hacerlo, o porque parezca mejorar mucho el sistema: el 90% de las funcionalidades adicionales no llegan a usarse. No prever requerimientos futuros ni flexibilidad extra. Concentrarse en el desarrollo de hoy.

Refactorizar a todo trance.

a) Refactorizar es mejorar el código existente, la estructura interna del software, no su comportamiento visible.
b) Refactorizar para mantener el diseño simple: quitar redundancia, eliminar funcionalidad no usada, rehacer diseños obsoletos, mantener el código limpio y conciso para que sea fácil de entender, modificar y extender.

El cliente siempre presente.

a) El cliente debe participar activamente a lo largo de todo el proceso, el experto, no un ayudante. Debe haber una pareja de representantes del cliente asignadas al proyecto, a veces en tiempo completo.
b) Los clientes escriben los relatos de usuario, participan en las estimaciones, asignan prioridades, verifican el cumplimiento de las funcionalidades. En la

reunión de planificación de entrega negocian el conjunto de relatos de usuario a implementar en cada entrega.

c) Los relatos de usuario no contienen detalle; los clientes deben estar presentes al definir las tareas de programación necesarias para implementar cada relato.
d) Los clientes deben estar presentes en las pruebas de verificación para revisar el resultado y determinar cuando puede pasarse a producción el sistema.

Normas de codificación.

a) Debe elegirse y respetarse una norma de codificación.

Codificar primero la prueba de unidad ("Unit Test").

a) Las pruebas de unidad se escriben una vez y se corren reiteradamente a lo largo de todo el proyecto, asegurando siempre el funcionamiento correcto; evitan las ambigüedades, los requerimientos quedan duros en la prueba.
b) Para cada unidad, se codifica primero una prueba simple para una función simple; se van agregando prueba y funcionalidad en etapas sucesivas, hasta implementar todo y probar todo en esa unidad.
c) Una funcionalidad está terminada cuando pasa todas sus pruebas de unidad.

Programación en parejas.

a) Todo el código incluido en una entrega de producción es escrito por dos personas trabajando juntas ante un único computador. Uno digita y piensa tácticamente sobre el método en construcción; el otro piensa estratégicamente en la integración de ese método en la clase.
b) La programación en pareja produce la misma cantidad de código a un nivel de calidad superior.
c) Lleva tiempo acostumbrarse a programar en pareja; puede resultar incómodo al principio.

Integración secuencial.

a) Los grupos trabajan en paralelo, pero integran uno por vez: sólo una pareja de programadores integra, prueba y entrega los cambios en el repositorio de código, en un momento dado. Allí se fija una nueva versión.
b) Una forma de asegurar la integración secuencial es hacerla en una máquina única (si los programadores trabajan en un mismo lugar físico).
c) Los programadores pueden integrar libremente en sus máquinas de trabajo habituales.

Integración frecuente.

a) Los programadores deben integrar su código al repositorio común cada pocas horas, al menos una vez por día cada pareja. Todos deben trabajar con la última versión.
b) La integración continua evita los esfuerzos divergentes o fragmentados donde no se comunica lo que se puede compartir o reutilizar; las incompatibilidades se detectan temprano.

Código de propiedad colectiva.

a) No hay una única persona responsable por el código: la construcción y la responsabilidad están distribuidas, todos los desarrolladores pueden agregar funcionalidad, corregir errores o refactorizar.
b) El código incluye siempre las pruebas de unidad; todos los cambios serán controlados por la suite de prueba. El código sólo puede entregarse luego de haber pasado el 100 % de las pruebas de unidad.

Optimizar al final.

a) Dejar la optimización para el final. No suponer dónde estarán los puntos de estrangulamiento; medirlos.

b) "Make it work, make it right, then make it fast." (Hágalo funcionar, hágalo bien, después hágalo andar rápido).

No trabajar horas extra.

a) El trabajo en horario extendido desmoraliza al equipo; no se cubrirá un atraso trabajando horas extra. Renegociar el alcance o la fecha de entrega en una reunión de planificación.
b) Agregar personal a un proyecto atrasado es una mala idea.

Pruebas de unidad.

a) Crear u obtener un marco de prueba ("test framework") para crear una suite automática de pruebas de unidad. Probar todas las clases del sistema (métodos triviales de "set" y "get" suelen omitirse).
b) Crear las pruebas antes de escribir el código. No puede integrarse código sin sus pruebas de unidad.
c) Las pruebas de unidad evolucionan junto con el código. No pueden crearse al final, ni dejar de escribirse.
d) El tiempo de escribir las pruebas de unidad se gana con creces en la reiteración continua de las pruebas y la confianza al encarar cambios.
e) Las pruebas de unidad posibilitan la propiedad colectiva de código, la refactorización, la integración frecuente. El agregado de funcionalidad incluye el agregado de pruebas.

Marco de prueba.

a) El marco de prueba no es una herramienta de prueba sino de desarrollo, se debe usar desde el principio. Puede crearse desde cero, pero los hay disponibles para la mayoría de los lenguajes (http://www.xprogramming.com/software.htm).

Cuando aparece un error.

a) Se crean pruebas para ese error, para evitar su reaparición.
b) La aparición de un error en producción requiere crear una prueba de aceptación. Ante una prueba de aceptación fallida, los programadores crean pruebas de unidad para ubicar el defecto en el código. Una vez superadas el 100% de las pruebas de unidad se vuelve a correr la prueba de aceptación fallida para verificar la desaparición del error.

Prueba de aceptación.

a) Las pruebas de aceptación se crean a partir de los relatos de usuario. El cliente define los escenarios de prueba para verificar si el relato de usuario ha sido correctamente implementado. Un relato de usuario puede tener una o varias pruebas de aceptación. El cliente es responsable de verificar el pasaje de las pruebas de aceptación y priorizar la corrección de las pruebas fallidas.
b) Las pruebas de aceptación son pruebas tipo caja negra a nivel del sistema: cada prueba de aceptación corresponde a un resultado producido por el sistema.
c) Las pruebas de aceptación deben ser automáticas, correrse frecuentemente, publicarse sus resultados y programarse su corrección para la próxima iteración.
d) Deben crearse pruebas de aceptación en cada iteración. Si no hay pruebas de aceptación nuevas no se ha hecho nada nuevo.
e) Un relato de usuario no está completo hasta no haber pasado todas sus pruebas de aceptación.

¿Qué propone XP?

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes de que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.

Esta metodología promueve los siguientes valores:

- Comunicación

El extreme Programming se nutre del ancho de banda más grande que se puede obtener cuando existe algún tipo de comunicación: la comunicación directa entre personas. Es muy importante entender cuáles son las ventajas de este medio. Cuando dos (o más) personas se comunican directamente pueden no sólo consumir las palabras formuladas por la otra persona, sino que también aprecian los gestos, miradas, etc. que hace su compañero. Sin embargo, en una conversación mediante el correo electrónico, hay muchos factores que hacen de esta una comunicación, por así decirlo, mucho menos efectiva.

- Coraje

El coraje es un valor muy importante dentro de la programación extrema. Un miembro de un equipo de desarrollo extremo debe de tener el coraje de exponer sus dudas, miedos, experiencias sin "embellecer" éstas de ninguna de las maneras. Esto es muy importante ya que un equipo de desarrollo extremo se basa en la confianza para con sus miembros. Faltar a esta confianza es una falta más que grave.

- Simplicidad

Dado que no se puede predecir cómo va a ser en el futuro el software que se está desarrollando; un equipo de programación extrema intenta mantener el software lo más sencillo posible. Esto quiere decir que no se va a invertir ningún esfuerzo en hacer un desarrollo que en un futuro pueda llegar a tener valor. En el XP frases como "...en un futuro vamos a necesitar..." o "Haz un sistema genérico de..." no tienen ningún sentido ya que no aportan ningún valor en el momento.

- Feedback

La agilidad se define (entre otras cosas) por la capacidad de respuesta ante los cambios que se van haciendo necesarios a lo largo del camino. Por este motivo uno de los valores que nos hace más ágiles es el continuo seguimiento o feedback que recibimos a la hora de desarrollar en un entorno ágil de desarrollo. Este feedback se toma del cliente, de los miembros del equipo, en cuestión de todo el entorno en el que se mueve un equipo de desarrollo ágil.

Características de XP, la metodología se basa en:

- **Pruebas Unitarias:** se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.

- **Refabricación:** se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

- **Programación en pares:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

Lenguajes de programación.

Uno de los ejes fundamentales que diferencian a Internet de otros medios de comunicación es la interacción y personalización de la información con el usuario. Esto se logra por medio de algunos de los diferentes lenguajes de programación para Web que existen en la actualidad. Dichos lenguajes se clasifican en dos partes fundamentales que reconocen la propia arquitectura Cliente/Servidor de esta

plataforma de desarrollo: los lenguajes del lado del Servidor y los lenguajes del lado del Cliente.

Entre los lenguajes del lado del Servidor podemos encontrar, entre los más sobresalientes por el auge que han tenido, algunos como PERL, ASP, PHP, Java, JSP, los módulos CGIs e ISAPIs, etc. Estos se caracterizan por desarrollar la lógica de negocio dentro del Servidor, además de ser los encargados del acceso a Bases de Datos y Tratamiento de la Información.

Del lado del Cliente se encuentran principalmente el JavaScript, el Visual Basic Script y el HTML, los que se encargan de facilitar una interfaz así como de solicitar y mostrar las consultas y procedimientos necesarios a la programación lógica.

Esta distinción en los lenguajes ha sido necesaria debido a que la Web funciona en modo “Desconectado”, o sea, un usuario a través de un navegador hace una petición de una página Web a un Servidor Web (Request), el Servidor recepciona la petición, la procesa y le envía la Respuesta al Cliente(Response), este la recibe y se desconecta.

A continuación se abordan las características fundamentales de algunos de estos lenguajes.

Lenguajes del lado del cliente.

HTML

Lenguaje de Marcas de Hipertexto (Hypertext Markup Language) por sus siglas en inglés, es el lenguaje de marcado predominante para la construcción de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores Web y otros procesadores de HTML.

HTML también es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma

descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores).

HTML consiste de varios componentes vitales, incluyendo elementos y sus atributos, tipos de data, y la declaración de tipo de documento.

JAVA SCRIPT

Es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

Al igual que Java, JavaScript es un lenguaje orientado a objetos propiamente dicho, ya que dispone de Herencia, si bien esta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas Web. Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del DOM.

Este lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, que es la que desarrolló los primeros navegadores Web comerciales. Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.

Tradicionalmente, se venía utilizando en páginas Web HTML, para realizar tareas y operaciones en el marco de la aplicación únicamente cliente, sin acceso a funciones del servidor. JavaScript se ejecuta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Los autores inicialmente lo llamaron Mocha y más tarde Live Script pero fue rebautizado como JavaScript en un anuncio conjunto entre Sun Microsystems y Netscape, el 4 de diciembre de 1995. JavaScript se puede incluir en cualquier documento HTML, o todo aquel que termine traduciéndose en HTML en el navegador del cliente; ya sea PHP, ASP, SVG... El código va inscrito dentro de los elementos HTML <script> y </script> se apoya su propuesta.

Lenguajes del lado del servidor.

PERL

Es un lenguaje de programación muy utilizado para construir aplicaciones CGI para el Web. Perl es un acrónimo de Practical Extracting and Reporting Language, que viene a indicar que se trata de un lenguaje de programación muy práctico para extraer información de archivos de texto y generar informes a partir del contendido de los ficheros.

Es un lenguaje libre de uso, eso quiere decir que es gratuito. Antes estaba muy asociado a la plataforma Unix, pero en la actualidad está disponible en otros sistemas operativos como Windows. Perl es un lenguaje de programación interpretado, al igual que muchos otros lenguajes de Internet como JavaScript o ASP.

JSP

JSP es un acrónimo de Java Server Pages, que traducido es algo así como Páginas de Servidor Java. Es una tecnología orientada a crear páginas Web con programación en Java.

Con JSP podemos crear aplicaciones Web que se ejecuten en variados servidores Web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma.

Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. Por tanto, las JSP podremos escribirlas con nuestro editor HTML/XML habitual.

PHP

PHP (Personal Home Page) es el acrónimo de Hypertext Preprocessor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Es también un lenguaje interpretado y embebido en el HTML.

Fue creado originalmente en 1994 por Rasmus Lerdorf, pero como PHP está desarrollado en política de código abierto, a lo largo de su historia ha tenido muchas contribuciones de otros desarrolladores. Actualmente PHP se encuentra en su versión 5, que utiliza el motor Zend, desarrollado con mayor meditación para cubrir las necesidades de las aplicaciones Web actuales.

PHP es un lenguaje de programación de estilo clásico, esto quiere decir que es un lenguaje de programación con variables, sentencias condicionales, bucles, funciones, etc. No es un lenguaje de marcas como podría ser HTML, XML o WML.

A diferencia de Java o JavaScript que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso permite acceder a los recursos que tenga el servidor, como por ejemplo podría ser, una base de datos. El programa PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML pero igualmente podría ser una página WML.

PHP es la gran tendencia en el mundo de Internet. Últimamente se puede observar un ascenso imparable, puesto que cada día son muchísimas más las páginas Web que lo utilizan para su funcionamiento, según las estadísticas, PHP se utiliza en más de 10 millones de páginas, y cada mes realiza un aumento del 15%. Como síntesis, PHP corre en 7 plataformas, funciona en 11 tipos de servidores, ofrece soporte sobre unas 20 Bases de Datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite; lo cual permite la creación de aplicaciones Web muy robustas, y contiene unas 40 extensiones estables sin contar las que se están experimentando, también tiene soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros. Además de que:

Es software libre, lo que implica menos costes y servidores más baratos que otras alternativas.

Es muy rápido. Su integración con la base de datos MySQL y el servidor Apache, le permite constituirse como una de las alternativas más atractivas del mercado.

Su sintaxis está inspirada en C, ligeramente modificada para adaptarlo al entorno en el que trabaja, de modo que si se está familiarizado con esta sintaxis, resultará muy fácil aprender PHP.
Su librería estándar es realmente amplia, lo que permite reducir los llamados "costes ocultos", uno de los principales defectos de ASP.
PHP tiene una de las comunidades más grandes en Internet, esto permite encontrar fácilmente ayuda, documentación, artículos, noticias, y otros recursos.
Permite las técnicas de Programación Orientada a Objetos.
No requiere definición de tipos de variables ni manejo detallado del bajo nivel.[8]

Framework de desarrollo

Codeigniter

- Se encuentra bajo la licencia open source Apache/BSD-style.
- Verdaderamente Liviano. El núcleo del sistema sólo requiere unas pocas pequeñas librerías. Esto es en duro contraste a muchos entornos de trabajo que requieren significativamente más recursos.
- Las librerías adicionales son cargadas dinámicamente a pedido, basado en sus necesidades para un proceso dado, así que el sistema base es muy delgado y bastante rápido.
- Usa el acercamiento Modelo-Vista-Controlador, que permite una buena separación entre lógica y presentación. Esto es particularmente bueno para proyecto en los cuales diseñadores están trabajando con sus archivos de plantilla, ya que el código en esos archivos será mínimo.
- Las URLs generadas por CodeIgniter son limpias y amigables a los motores de búsqueda. En vez de usar el acercamiento estándar "query string" a las

[8] Los detalles de los lenguajes ya sea del lado del servidor, como los del lado del cliente fueron tomados de:
La enciclopedia en línea Wikipedia.

URLs que es sinónimo de sistemas dinámicos, CodeIgniter usa un acercamiento basado en segmentos.

- Viene con un rango lleno de librerías que le permiten realizar las tareas de desarrollo web más comúnmente necesarias, como acceder a una base de datos, mandar un email, validar datos de un formulario, mantener sesiones, manipular imágenes, trabajando con datos XML-RPC y mucho más.
- El sistema puede ser fácilmente extendido a través del uso de plugins y librerías asistentes, o a través de extensión de clases o ganchos del sistema.
- Aunque CodeIgniter si viene con un motor de plantillas simple que puede ser opcionalmente usado, no le fuerza a usarlo. Los motores de plantilla simplemente no pueden igualar el desempeño del nativo PHP, y la sintaxis que debe ser aprendida para usar un motor de plantilla es más fácil que aprender la base de PHP.

1.5 Conclusiones

Partiendo del estudio del tema que nos interesa, los problemas y necesidades y de las características existentes, se brindan detalles de las herramientas que serán utilizadas para la culminación de este trabajo, con los detalles anteriormente dados, las explicaciones de funcionalidad y los pasos a seguir para la automatización de esta herramienta podemos decir que está creada la base para la realización del presente trabajo de diploma.

Capítulo 2

CARACTERÍSTICAS DEL SISTEMA

Introducción

En el presente capítulo se describen los parámetros principales que constituyen las entradas y las salidas de datos del sistema, que constituyen el punto de partida para la construcción de las interfaces de entrada y salida del mismo. Se presenta, de la primera fase de la metodología, las historias de usuarios realizadas por el cliente, así como toda la planificación de entrega para su implementación.

2.1 Especificaciones de configuración para los datos de entrada

En el sistema los datos de entrada se realizan manualmente definiéndose los siguientes datos que serán aceptados por el sistema, el cual podrá mantener y procesar información de un proyecto en desarrollo:

1) Los datos del proyecto tendrán el formato siguiente: su nombre, el problema al cual responde, causa del problema, efecto del mismo.
2) Pueden existir x cantidad de objetivos a solucionar en un mismo proyecto, esto define que la solución al problema sea la más adecuada.
3) Lo mismo sucede con las actividades, donde un objetivo de un proyecto tendrá cuantas actividades requiera y el incumplimiento de alguna no es un impedimento para dar un proyecto por terminado.
4) El propósito no es más que la justificación de la actividad a cumplir.
5) Cada elemento tiene una prioridad para de esta forma definir si es indispensable para el cumplimiento.

2.2 *Personas relacionadas con el sistema.*

Persona(s) relacionada(s) con el sistema.	***Justificación***
Especialistas	Estas son las personas que tienen cierto conocimiento en materia de organización y proyección en la creación de proyectos, y están encargados de introducir los datos iniciales para la creación de un proyecto en el sistema y de la selección correcta de los reportes necesarios.

Tabla 1: Personas relacionadas con el sistema.

2.3 *Historias de usuarios*

La historia de usuario es la técnica utilizada en la metodología XP para la especificidad de los requisitos del software. Es el resultado directo del intercambio entre los usuarios y los desarrolladores a través de reuniones; mediante el diálogo y el intercambio de ideas en dichas reuniones se conocerán los requerimientos del sistema, las posibles soluciones y se definirán cuáles son las historias más importantes. En general se describen brevemente las características que el sistema debe tener desde la perspectiva del cliente.

Historia de usuario	
Numero: (número de la historia de usuario)	***Usuario:*** (usuario entrevistado para obtener los requisitos del sistema)
Nombre de la historia: (Nombre de la historia de usuario que sirve para identificarla mejor entre los desarrolladores y el cliente)	
Prioridad en negocios: (Importancia de la historia para el cliente) (Alta / Media / Baja)	***Riesgo en desarrollo:*** (Dificultad para el programador) (Alto / Medio / Bajo)
Puntos estimados: *2*	***Iteración asignada:*** (Iteración a la que corresponde)
Programador responsable: Yoangel Hernández García.	

Descripción: (Se especifican las operaciones por parte del usuario y las respuestas que dará el sistema)
Observaciones: (Algunas observaciones de interés, como glosario, información sobre usuarios, etc.)

- Historias de usuarios definidas para este sistema.

Historia de usuario	
Numero: **1**	***Usuario:*** **Delmis Wilfredo Carbonell Laborde**
Nombre de la historia: Introducción de datos al sistema.	
Prioridad en negocios: Medio	***Riesgo en desarrollo:*** Bajo
Puntos estimados: 2	***Iteración asignada:*** **1**
Programador responsable: Yoangel Hernández García.	
Descripción: Los usuarios definidos por los jefes de proyectos y el administrador serán capaces de introducir los datos correspondientes a cada proyecto.	
Observaciones: no se puede tener fallos al introducir los datos al sistema.	

Tabla: Historia de usuario: Introducción de datos al sistema.

Historia de usuario	
Numero: **2**	***Usuario:*** **Delmis Wilfredo Carbonell Laborde**
Nombre de la historia: Presentación de los reportes que generará el sistema.	
Prioridad en negocios: Alto	***Riesgo en desarrollo:*** Medio
Puntos estimados: 4	***Iteración asignada:*** **2**
Programador responsable: Yoangel Hernández García.	
Descripción: Los usuarios del sistema son los que le harán la solicitud al sistema.	
Observaciones: El sistema procesará la petición definida por el usuario y le devolverá la solicitud realizada.	

Tabla 2: Historia de usuario: Presentación de los reportes que generará el sistema.

Historia de usuario	
Numero: **3**	***Usuario:*** **Berto Andrés Martín.**
Nombre de la historia: Presentación de la matriz de marco lógico.	
Prioridad en negocios: Alta	***Riesgo en desarrollo:*** Alto
Puntos estimados: 5	***Iteración asignada: 3***
Programador responsable: Yoangel Hernández García.	
Descripción: El sistema según el usuario que haga la solicitud de presentar la matriz de un proyecto dado y los privilegios del mismo definirá si puede o no.	
Observaciones: Un mismo usuario puede tener derechos sobre varios proyectos, por lo que el usuario será el que seleccione el proyecto al cual quiere realizar la matriz.	

Tabla 3: Historia de usuario: Presentación de la matriz de marco lógico.

2.4 Planificación de entregas

Para la plantación de la entrega se establece la prioridad de cada historia de usuario así como una estimación del esfuerzo necesario de cada una de ellas con el fin de determinar un cronograma de entregas.

Las estimaciones de esfuerzo asociado a la implementación de las historias se establecen utilizando como medida el punto. Un punto, equivale a una semana ideal de programación que en este caso seria de 5 días. Las historias generalmente valen de 1 a 5 puntos. Por otra parte, se mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de

historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

2.5.1 *Estimación de esfuerzos por historias de usuario.*

Historia de usuario	*Número*	*Puntos estimados*
❖ Introducción de datos al sistema	**1**	**2**
❖ Presentación de los reportes que generará el sistema	**2**	**4**
❖ Presentación de la matriz de marco lógico	**3**	**5**

Tabla 4: Estimación de esfuerzos por historias de usuario.

El plan de entregas se realiza teniendo en cuenta las unidades funcionales que se quieren entregar y cada uno de estos módulos abarca un número de historias de usuarios a implementar para dar cumplimiento al funcionamiento del mismo.

2.5.2 Plan de entrega

Módulos	*Historia(s) de Usuario que abarca*
❖ Generación de los reportes que tendrá el sistema	***1,2***
❖ matriz de marco lógico	***3***

Tabla 5: Plan de entrega.

2.5.3 Plan de duración de la Iteración.

Teniendo en cuenta las Historias de usuarios anteriormente presentadas se realizará una planificación en tres iteraciones basándonos en el tiempo y procurando obtener la funcionalidad relacionada en la misma iteración.

Iteración	*Orden de implementación por Historias de Usuario*	*Duración total de la iteración en semanas*
1	❖ Introducción de datos al sistema	2
2	❖ Presentación de los reportes que tendrá el sistema	4
3	❖ Presentación de la matriz de marco lógico	5

Tabla 6: Duración de las iteraciones.

Combinando el plan de entrega y el plan de iteraciones se harán releases o liberaciones al sistema en las fechas mostradas a continuación:

Iteración \ Módulo	***Generación de los reportes que tendrá el sistema***	***Matriz de marco lógico.***
Fin 1ra iteración	16 marzo 2010	
Fin 2da iteración	13 abril 2010	
Fin 3ra iteración		18 mayo 2010

Tabla 7: Tabla de releases.

2.6 *Conclusiones*

La entrada de los datos al sistema, la generación de reportes, la gestión y presentación de la matriz de marco lógico, junto a las historias de usuarios creadas por el cliente constituyen el objeto de automatización de la presente investigación. A partir de las historias de usuarios se planifica la entrega del software, dicha entrega se realiza conjuntamente con el cliente que es el actor principal en la etapa de planificación de la metodología seleccionada. Se decide realizar el software en tres iteraciones de 2,4 y 5 semanas de trabajo cada una para garantizar la entrega al cliente en el tiempo establecido.

Capítulo 3

DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

3.1 Introducción

En este capítulo se presentan las fases de diseño e implementación de la metodología XP. Uno de los artefactos principales es la creación de las tarjetas CRC (Clase-Responsabilidades-Colaboración) las cuales permiten brindar un mayor enfoque orientado a objetos. Por otra parte se describen cada una de las tareas confeccionadas para cumplir con el desarrollo de cada una de las historias de usuario detectadas.

3.2 Modelo Vista Controlador (MVC)

Implementado bajo el framework de desarrollo Codeigniter, es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

- ✓ **Modelo**: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.
- ✓ **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- ✓ **Controlador**: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta in dirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

3.3 Tarjetas CRC

El uso de las tarjetas CRC (Class, Responsibilities and Collaboration) permiten al programador centrarse y apreciar el desarrollo orientado a objetos, olvidándose de los malos hábitos de la programación procedural clásica.

Las tarjetas CRC representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden

escribir las responsabilidades u objetivos que debe cumplir el objeto y, a la derecha, las clases que colaboran con cada responsabilidad.

Esta nueva técnica de diseño es adoptada como alternativa a los diagramas UML de las clases, pues en estas se plasman las responsabilidades que tiene cada objeto,así como las clases con las que tienen que interactuar para darles respuesta, brindando de esta manera la información que se necesita a la hora de implementar.

Clase	
Responsabilidades	**Colaboraciones**
Clase A	Clase B

Tabla 8: Plantilla de Tarjeta CRC.

3.3.1 *Tarjetas CRC del modulo #1: Generación de los reportes que tendrá el sistema.*

Gestión de los usuarios del sistema	
Responsabilidades	**Colaboraciones**
Insertar usuarios	Control usuarios
Editar usuarios	
Eliminar usuarios	
Relacionar un usuario con un proyecto	

Tabla 9: Tarjeta CRC de Gestión de los usuarios del sistema.

Gestión de los involucrados del sistema	
Responsabilidades	**Colaboraciones**
Insertar involucrado	Control involucrado
Editar involucrado	
Eliminar involucrado	
Relacionar un involucrado con un proyecto	

Tabla 10: Tarjeta CRC de Gestión de los usuarios del sistema.

Control de proyectos	
Responsabilidades	**Colaboraciones**
Insertar proyecto	Gestión de proyecto
Editar proyecto	
Eliminar proyecto	
Visualizar contenido del proyecto	Análisis de proyecto
Análisis de un proyecto	
Relacionar un proyecto con varios involucrados	

Tabla 11: Tarjeta CRC de Control de proyectos.

Control de objetivos correspondientes a un proyecto	
Responsabilidades	**Colaboraciones**
Insertar objetivos	Gestión de objetivos
Editar objetivos	
Eliminar objetivos	
Visualizar contenido de los objetivos	

Tabla 12: Tarjeta CRC de Control de objetivos correspondientes a un proyecto.

Control de actividades correspondientes a un objetivo	
Responsabilidades	**Colaboraciones**
Insertar actividades	Gestión de actividades
Editar actividades	
Eliminar actividades	
Visualizar contenido de las actividades	

Tabla 13: Tarjeta CRC de Control de actividades correspondientes a un objetivo.

Control de propósitos correspondientes a una actividad	
Responsabilidades	**Colaboraciones**
Insertar propósito	Gestión de propósito
Editar propósito	
Eliminar propósito	
Visualizar contenido del propósito	

Tabla 14: Tarjeta CRC de Control de propósitos correspondientes a una actividad.

Gestión de reportes del sistema	
Responsabilidades	**Colaboraciones**
Visualizar el contenido de un proyecto	Gestión de reportes
Generación de la Matriz de Marco Lógico	
Reporte de las actividades de un proyecto y los involucrados relacionados a esas actividades	
Reporte del cumplimiento progresivo del proyecto	

Tabla 15: Tarjeta CRC de Gestión de reportes del sistema.

3.3.2 *Tarjetas CRC del modulo #2: matriz de marco lógico.*

Gestión de las actividades del sistema	
Responsabilidades	**Colaboraciones**
Análisis de las actividades	Análisis de Actividades de un proyecto
Análisis del propósito a cumplir por parte de cada actividad	

Tabla 16: Tarjeta CRC de Gestión de las actividades del sistema.

Gestión de los propósitos del sistema	
Responsabilidades	**Colaboraciones**
Análisis de los propósitos	Análisis de propósitos de un proyecto
Análisis de las actividades a cumplir por parte de cada propósito	

Tabla 17: Tarjeta CRC de Gestión de los propósitos del sistema.

Gestión de elementos del sistema	
Responsabilidades	**Colaboraciones**
Análisis del proyecto	Gestión de Elementos
Análisis de Actividades de un proyecto	
Análisis de propósitos de un proyecto	

Tabla 18: Tarjeta CRC de Gestión de Elementos.

3.4 Implementación

La metodología XP se convierte en una integrante más del equipo de desarrollo, el cliente crea las historias de usuario bajo la supervisión de los desarrolladores. Estas historias quedan confeccionadas cuando el cliente es capaz de identificar con precisión la funcionalidad deseada, además, también debe estar presente cuando se realicen las pruebas de aceptación para cada historia, por lo que su presencia es imprescindible.

En XP generalmente cada historia de usuario se divide en tareas de ingeniería (TI) o tareas de programación. Estas se crean para obtener una mejor planificación de la historia; con ellas se pretende cumplir con las funcionalidades básicas que luego conformarán las funcionalidades generales de cada historia.

3.5. Tareas por historias de usuarios.

Tareas de la historia de usuarios #1:

	Tarea
Número tarea: 1	***Número historia:*** 1
Nombre de la tarea: Creación de la interfase para verificación y obtención de los datos de entrada del sistema.	
Tipo de tarea: Desarrollo	***Puntos estimados:*** 1.5
Fecha de inicio: 2 marzo 2010	***Fecha de fin:*** 12 marzo 2010
Programador responsable: Yoangel Hernández García	
Descripción: Identificar los datos que tendrá el sistema.	

Tabla 19 THU: Creación de la interfase para verificación y obtención de los datos de entrada del sistema.

	Tarea
Número tarea: 2	***Número historia:*** 1
Nombre de la tarea: Verificación de los datos de entrada del sistema.	
Tipo de tarea: Desarrollo	***Puntos estimados:*** 0.5
Fecha de inicio: 15 marzo 2010	***Fecha de fin:*** 16 marzo 2010
Programador responsable: Yoangel Hernández García	
Descripción: Se identifican los datos de entrada que se le darán a las variables para a la hora de insertar en el sistema de base de datos corresponda con el tipo que tiene en la misma.	

Tabla 20 THU: Verificación de los datos de entrada del sistema.

Tareas de la historia de usuarios #2:

	Tarea
Número tarea: 1	***Número historia:*** 2
Nombre de la tarea: Reporte de la situación de un proyecto	
Tipo de tarea: Desarrollo	***Puntos estimados:*** 1
Fecha de inicio: 17 marzo 2010	***Fecha de fin:*** 24 marzo 2010
Programador responsable: Yoangel Hernández García	
Descripción: Se describe la situación hasta el momento que se solicita el reporte del proyecto al sistema con las actividades cumplidas y las no cumplidas.	

Tabla 21 THU: Reporte de la situación de un proyecto.

	Tarea
Número tarea: 2	***Número historia:*** 2
Nombre de la tarea: Reporte de la matriz de un proyecto	

Tipo de tarea: Desarrollo	***Puntos estimados:*** 1
Fecha de inicio 25 marzo 2010	***Fecha de fin:*** 2 abril de 2010
Programador responsable: Yoangel Hernández García	
Descripción: este reporte dará información de in proyecto dado y generara en un formato PDF y con forma de matriz los elementos obtenidos de ese proyecto.	

Tabla 22 THU: Reporte de la matriz de un proyecto.

Tarea	
Número tarea: 3	***Número historia:*** 2
Nombre de la tarea: Reporte de las actividades y sus involucrados en un proyecto	
Tipo de tarea: Desarrollo	***Puntos estimados:*** 1
Fecha de inicio 5 abril de 2010	***Fecha de fin:*** 9 abril de 2010
Programador responsable: Yoangel Hernández García	
Descripción: teniendo solo la información del proyecto se podrán saber las actividades a realizar y los encargados de llevarlas a cabo, o sea los involucrados y si se cumplió o no cada una de las actividades.	

Tabla 23 THU: Reporte de las actividades y sus involucrados en un proyecto.

Tarea	
Número tarea: 4	***Número historia:*** 2
Nombre de la tarea: Reporte del cumplimiento progresivo del proyecto	
Tipo de tarea: Desarrollo	***Puntos estimados:*** 1
Fecha de inicio 12 abril de 2010	***Fecha de fin:*** 16 abril de 2010
Programador responsable: Yoangel Hernández García	
Descripción: llevando a cabo un análisis de las actividades definidas por el especialista para un proyecto se podrá conocer cual es el nivel de cumplimiento de un proyecto en el mismo instante de la realización de la solicitud.	

Tabla 24 THU: Reporte del cumplimiento progresivo del proyecto.

Tareas de la historia de usuarios #3:

Tarea	
Número tarea: 1	***Número historia:*** 3
Nombre de la tarea: Interfase de trabajo para la generación y muestra de los elementos correspondiente a la matriz de marco lógico.	
Tipo de tarea: Desarrollo	***Puntos estimados:*** 1.5
Fecha de inicio: 14 abril de 2010	***Fecha de fin:*** 23 abril de 2010
Programador responsable: Yoangel Hernández García	
Descripción: Con la creación de esta interfase se podrá visualizar el contenido de cada proyecto y se mostrarán los elementos que componen para un proyecto la matriz.	

Tabla 25 THU: Interfase de trabajo para la generación y muestra de los elementos correspondientes a la matriz de marco lógico.

	Tarea
Número tarea: 2	***Número historia:*** 3
Nombre de la tarea: Gestión de los elementos correspondientes a un proyecto presente en el sistema.	
Tipo de tarea: Desarrollo	***Puntos estimados:*** 1.5
Fecha de inicio: 26 abril de 2010	***Fecha de fin:*** 4 mayo 2010
Programador responsable: Yoangel Hernández García	
Descripción: Verifica y gestiona a la hora de general el principal recurso funcional del sistema que es la matriz de marco lógico en el cual se obtendrán los datos e informaciones correspondientes al proyecto existente.	

Tabla 26 THU: Gestión de los elementos correspondientes a un proyecto presente en el sistema.

	Tarea
Número tarea: 3	***Número historia:*** 3
Nombre de la tarea: Obtener y analizar las actividades correspondientes a un proyecto.	
Tipo de tarea: Desarrollo	***Puntos estimados:*** 2
Fecha de inicio: 5 mayo 2010	***Fecha de fin:*** 18 mayo 2010
Programador responsable: Yoangel Hernández García	
Descripción: En esta parte se obtendría cada actividad correspondiente a un proyecto y se podría saber a quien de los involucrados en el proyecto es el encargado de llevar a cabo dicha actividad, y saber si se cumplió la misma.	

Tabla 27 THU: Obtener y analizar las actividades correspondientes a un proyecto.

3.6 Conclusiones

Enfocándose en la programación orientada a objetos utilizada en el sistema dentro de la fase de diseño de la metodología XP[9], se crearon las tarjetas CRC. Para lograr la completa implementación de cada historia de usuario en la fecha acordada con el cliente, estas se dividieron en tareas de ingeniería. A cada Tarea de Historia de Usuario se le asignó un tiempo de desarrollo cumpliéndose de manera eficiente garantizando así el objetivo principal de su confección.

[9] XP Extreme Programming (Programación extrema).

Capítulo 4

PRUEBA

Introducción

En el presente capítulo se muestran las pruebas de aceptación confeccionadas por el cliente para comprobar que el sistema funcione de forma correcta. Estas pruebas se fueron llevando a cabo antes de cada entrega de los componentes del sistema que se realizaron durante todo el desarrollo del proyecto.

4.1 Pruebas de aceptación.

Uno de las mejores características de la metodología XP es el proceso de pruebas. Esta metodología propone probar tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos no deseados a la hora de realizar modificaciones y refactorizaciones. XP propone la realización de pruebas unitarias, encargadas de verificar el código y diseñadas por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida por el cliente.

En este proyecto al codificar no se sigue la regla de XP que aconseja crear pruebas unitarias con entornos de desarrollo antes de programar. Las obtuvieron de la descripción de requisitos plasmados en las historias de usuarios, y estas especifican las barreras que deben pasar las distintas funcionalidades del programa, procurando codificar teniendo en cuenta las pruebas que se deben vencer. Para realizar las pruebas de aceptación el cliente utiliza la siguiente plantilla.

Prueba de aceptación
HU*:* Nombre de la historia de usuario que va a comprobar su funcionamiento.
Nombre: Nombre del caso de prueba.
Descripción: Descripción del propósito de la prueba.
Condiciones de ejecución: Precondiciones para que la prueba se pueda realizar.
Entrada/Pasos de ejecución: Pasos para probar la funcionalidad.
Resultado esperado: Resultado que se desea de la prueba.
Evaluación de la prueba: Aceptada o Denegada.

Tabla 28: Plantilla Prueba de Aceptación.

4.2 Prueba del Modulo #1: Generación de los reportes que tendrá el sistema.

Prueba de aceptación
HU: Introducción de datos al sistema.
Nombre: Verificación de datos al crear o modificar un usuario.
Descripción: La persona encargada del llenado de datos al sistema ingresa o modifica características de los usuarios.
Condiciones de ejecución: • A la hora de realizar el ingreso de un usuario no puede existir en la base de datos dicho usuario. • Llenado de los datos imprescindibles del sistema.
Entrada/Pasos de ejecución: para agregar, modificar o eliminar usuarios • Seleccione el tab. de usuarios. • Para el ingreso podrás seleccionar el botón **Agregar usuario**. • Para la modificación podrás seleccionar el botón **Modificar usuario**. • Para la eliminar podrás seleccionar el botón **Eliminar usuario**.
Resultado esperado: Se actualiza en la base de datos según lo que realice, si se eliminó un usuario entonces se eliminará la relación de ese usuario con un proyecto y el involucrado al cual le pertenece y se recargará la página mostrando que sea eliminado el usuario.
Evaluación de la prueba: Aceptada.

Tabla 29: Verificación de datos al crear o modificar un usuario.

Prueba de aceptación
HU: Introducción de datos al sistema.
Nombre: Verificación de datos al crear o modificar un involucrados.
Descripción: La persona encargada del llenado de datos al sistema ingresa o modifica características de los involucrados.
Condiciones de ejecución: • A la hora de realizar el ingreso de un involucrado no puede existir en la base de datos dicho involucrado. • Llenado de los datos imprescindibles del sistema.
Entrada/Pasos de ejecución: para agregar, modificar o eliminar involucrado • Seleccione el tab. de involucrados. • Para el ingreso podrás seleccionar el botón **Agregar involucrado**. • Para la modificación podrás seleccionar el botón **Modificar involucrado**. • Para la eliminar podrás seleccionar el botón **Eliminar involucrado**.
Resultado esperado: Se actualiza en la base de datos según lo que realice, si se elimino un involucrado entonces se eliminara la relación de ese involucrado con un proyecto y se recargará la página mostrando que se ha eliminado el involucrado.
Evaluación de la prueba: Aceptada.

Tabla 30: Verificación de datos al crear o modificar un involucrados.

Prueba de aceptación
HU: Introducción de datos al sistema.
Nombre: Verificación de datos al crear o modificar un proyecto.
Descripción: La persona encargada del llenado de datos al sistema ingresa o modifica características del proyecto.
Condiciones de ejecución: • A la hora de realizar el ingreso de un involucrado no puede existir en la base de datos dicho proyecto. • Llenado de los datos imprescindibles del sistema.
Entrada/Pasos de ejecución: para agregar, modificar o eliminar proyecto • Seleccione el tab. de proyectos. • Para el ingreso podrás seleccionar el botón **Agregar proyecto**. • Para la modificación podrás seleccionar el botón **Modificar proyecto**. • Para la eliminar podrás seleccionar el botón **Eliminar proyecto**.

Resultado esperado: Se actualiza en la base de datos según lo que realice, si se elimino un proyecto entonces se eliminara la relación de ese proyecto con los usuarios e involucrados relacionados con el y se recargara la pagina mostrando que se a eliminado el proyecto.
Evaluación de la prueba: Aceptada.

Tabla 31: Verificación de datos al crear o modificar un proyecto.

Prueba de aceptación
HU: Introducción de datos al sistema.
Nombre: Verificación de la entrada de datos de las actividades y elementos de un proyecto.
Descripción: La persona encargada del llenado de datos al sistema ingresa o modifica características del proyecto.
Condiciones de ejecución: Que pertenezca los elementos al proyecto correcto.
Entrada/Pasos de ejecución: para agregar, modificar o eliminar elementos • Seleccione el tab. de proyectos para agregar actividades a un proyecto. • Seleccione el tab. de actividades para eliminar actividades. • Seleccione el tab. de actividades para agregar actividades.
Resultado esperado: Se actualiza en la base de datos según lo que realice, si se eliminó una actividad entonces se eliminará la relación de esa actividad con el proyecto relacionados con él y se recargará la página mostrando que se ha eliminado la actividad.
Evaluación de la prueba: Aceptada.

Tabla 32: Verificación de la entrada de datos de las actividades y elementos de un proyecto.

Prueba de aceptación
HU: Presentación de los reportes que generará el sistema.
Nombre: Visualizar el contenido de un proyecto.
Descripción: El sistema según el contenido que se le solicite devolverá un reporte de lo pedido en este caso verificará el proyecto solicitado y generará un reporte del proyecto con todos los elementos del mismo.
Condiciones de ejecución: Seleccionar un proyecto y realizar la solicitud del reporte.
Entrada/Pasos de ejecución: Para generar un reporte de un proyecto.

• Seleccione el tab. Reportes para visualizar un conjunto de reportes. • Seleccionar el tipo de reporte que se desea (reporte de proyecto). • Generar el reporte del proyecto con sus actividades, objetivo, involucrado y demás elementos.
Resultado esperado: Se recopilan del sistema todos los elementos que conforman el proyecto y genera un reporte pudiéndolo exportar como un fichero PDF.
Evaluación de la prueba: Aceptada.

Tabla 33: Visualizar el contenido de un proyecto.

Prueba de aceptación
HU: Presentación de los reportes que generará el sistema.
Nombre: Reporte del cumplimiento progresivo del proyecto**.**
Descripción: El sistema según el contenido que se le solicite devolverá un reporte de lo pedido, en este caso verificará el proyecto solicitado y genera un reporte del proyecto y su por ciento de cumplimiento.
Condiciones de ejecución: Seleccionar un proyecto y realizar la solicitud del reporte.
Entrada/Pasos de ejecución: Para generar un reporte de un proyecto. • Seleccione el tab. Reportes para visualizar un conjunto de reportes. • Seleccionar el tipo de reporte que se desea (reporte de proyecto). • Generar el reporte del proyecto con sus actividades y el por ciento de cumplimiento del mismo según sus actividades.
Resultado esperado: Se recopilan del sistema todos los elementos que conforman el proyecto y genera un reporte de cumplimiento según las actividades que han sido realizadas pudiéndolo exportar como un fichero PDF.
Evaluación de la prueba: Aceptada.

Tabla 34: Reporte del cumplimiento progresivo del proyecto.

4.3 Prueba del Modulo #2: matriz de marco lógico.

Prueba de aceptación
HU: Presentación de la matriz de marco lógico.
Nombre: Presentación de la matriz de marco lógico.
Descripción: Según el usuario que solicite la presentación de la matriz de marco lógico se llevara a cabo o no, dependiendo de la seguridad.
Condiciones de ejecución: Seleccionar la presentación de la matriz y realizar la solicitud del reporte.
Entrada/Pasos de ejecución: Para generar la matriz. • Seleccionar el tab. matriz de marco lógico.
Resultado esperado: Se muestra la matriz de los proyectos existentes en el sistema.
Evaluación de la prueba: Aceptada.

Tabla 35: Presentación de la matriz de marco lógico.

Prueba de aceptación
HU: Presentación de la matriz de marco lógico.
Nombre: Gestión de los elementos correspondientes a un proyecto presente en el sistema.
Descripción: Según el usuario que solicite la lectura, o a la hora de modificar cualquier valor que tenga relación con elementos de un proyecto el sistema podrá verificar y ubicar cada elemento con su relación correspondiente.
Condiciones de ejecución:
Entrada/Pasos de ejecución: Para verificar los elementos de un proyecto. • Seleccionar los tab. En los que se muestran las relaciones entre los elementos del proyecto. • Se podrá realizar la relación de los mismos a través de estos tab.
Resultado esperado: Cada elemento que contenga una relación a otro elemento se le mostrará al usuario.
Evaluación de la prueba: Aceptada.

Tabla 36: Gestión de los elementos correspondientes a un proyecto presente en el sistema.

4.4 Conclusiones

Con la realización de las pruebas de aceptación el cliente se asegura de que las funciones implementadas cumplan su objetivo satisfactoriamente, probando individualmente cada módulo y asignándole la evaluación correspondiente. Todas las pruebas que se realizaron fueron positivas y el cliente estuvo conforme, cumpliendo entonces la aplicación con las historias de usuarios definidas inicialmente.

Capítulo 5

ESTUDIO DE FACTIBILIDAD

Introducción

Para la factibilidad de este proyecto se utilizará la **Metodología Costo Efectividad (Beneficio),** la cual plantea que la conveniencia de la ejecución de un proyecto se determina por la observación conjunta de dos factores:

- El costo, que involucra la implementación de la solución informática, adquisición y puesta en marcha del sistema hardware/software y los costos de operación asociados
- La efectividad, que se entiende como la capacidad del proyecto para satisfacer la necesidad, solucionar el problema o lograr el objetivo para el cual se ideó; es decir, un proyecto será más o menos efectivo con relación al mayor o menor cumplimiento que alcance en la finalidad para la cual fue ideado (costo por unidad de cumplimiento del objetivo).

5.1 Efectos Económicos

- Efectos directos
- Efectos indirectos
- Efectos externos
- Intangibles

Efectos directos:

POSITIVOS:

- Se facilitará la entrada de datos al sistema.
- Se facilitará la interpretación de los resultados mediante la matriz de marco lógico.

- Se logra una completa planificación y proyección de un proyecto a desarrollar.
- Eleva las acciones de control del líder o líderes del o los proyectos asociados, independientemente de donde se puedan encontrar los ejecutores, favoreciendo la toma de decisiones y evitando el incumplimiento de los plazos previstos para el desarrollo del software.
- Permite a la entidad planificadora, organizadora y ejecutora del proyecto, precisión en los actores que formarán parte del mismo, plantear sus costos de manera más detallada, vislumbrar una idea a partir del árbol de problemas y de objetivos de la presencia de las categorías filosóficas causa-efecto que se puedan presentar durante cualquiera de las etapas del proyecto y los consiguientes pronósticos de las soluciones a dar a cada una en particular.
- Su aplicación no contradice la actual metodología de desarrollo de software que tiene diseñada la empresa DESOFT S.A., División Guantánamo, por lo contrario, contribuye a su perfeccionamiento, al insertarse aspectos de las ciencias.

NEGATIVOS:

- Para usar la aplicación es imprescindible el uso de un ordenador como mínimo, aparejado a los gastos que este trae de consumo de energía eléctrica y mantenimiento.

Efectos indirectos:

- Inicialmente se pretende utilizar este software en la empresa DESOFT División Gtmo., aunque existen aspiraciones de comercializarlo – en el futuro – en el resto de las empresas del país.

Efectos Externos:

- Con este sistema se gana una herramienta informática para la buena planificación y control de proyectos en la empresa de Desarrollo de Software de Guantánamo.

***Intangibles*:**

En la valoración económica siempre hay elementos perceptibles por una comunidad como perjuicio o beneficio, pero al momento de ponderar en unidades monetarias esto resulta difícil o prácticamente imposible.
Con el fin de medir con precisión los efectos, deberán considerarse tres situaciones

1 Situación sin proyecto

Actualmente a la hora de diseñar un proyecto y plantearlo no se definen detalladamente todas la actividades y los elementos indispensables para llevar a cabo ese proyecto, y la existencia de una metodología a utilizar a la hora de dicha planificación por parte del CITMA en la provincia provoca que no se pueda desarrollar un control de los proyectos automatizados, que el factor tiempo aumenta al no contar con una herramienta informática que favorezca al proceso organizativo y de control del desarrollo del software por lo que se plantea la creación de esta herramienta que permitirá controlar y planificar los proyectos de la empresa.

2 Situación con proyecto

Se automatizará el proceso que hasta el momento se realizaba a mano y se controlaba mediante papeles. La entrada de datos se realizará por los especialistas y los usuarios encargados podrán ir dándole seguimiento a lo planificado en el proyecto a medida que se realice.

El o los líderes de los proyectos podrán contar con información para la toma de decisiones de forma actualizada y con precisión acerca de su estado actual.
Permite un grado de detalle acerca de la factibilidad de los involucrados en el proyecto y del costo real del mismo.

Permite que puedan ser relacionados varios proyectos al unísono, tal y como se manifiestan en la actualidad los diversos compromisos con que cuenta la empresa, tanto en el territorio como en el extranjero.

Permite a la empresa Desoft S.A. División Guantánamo el empleo de una metodología para el desarrollo del software a partir de los elementos de la vanguardia a nivel mundial, lo que les favorecerá ganar en competitividad en el mercado internacional.

5.2 Beneficios y costos intangibles en el proyecto

***COSTOS*:**

- ✓ Resistencia al cambio.
- ✓ Capacitación de los operadores en la metodología de Marco Lógico

***BENEFICIOS*:**

- ✓ Mayor comodidad para los usuarios.
- ✓ Mejora en la calidad de la información.
- ✓ Menor tiempo empleado en la introducción de los datos.
- ✓ Facilidad a la hora de interpretar los datos que genera el sistema.
- ✓ Agilidad y Oportunidad en la información para la toma de la decisión por parte del o los líderes de proyectos.

5.3 Ficha de Costo.

Para determinar el costo económico del proyecto se utilizará el procedimiento para elaborar Una Ficha De Costo de un Producto Informático [Dra. Ana Ma. Gracia Pérez, UCLV].

Para la elaboración de la ficha se consideran los siguientes elementos de costo, desglosados en moneda libremente convertible y moneda nacional.

Costos en Moneda Libremente Convertible:

- ✓ ***Costos Directos.***
  1. Compra de equipos de cómputo: No procede.
  2. Alquiler de equipos de cómputo: No procede.
  3. Compra de licencia de Software: No procede.
  4. Depreciación de equipos: $ 40.00.
  5. Materiales directos: No procede.

  Total: $ 40.00.
- ✓ ***Costos Indirectos.***
  1. Formación del personal que elabora el proyecto: No procede.
  2. Gastos en llamadas telefónicas: No procede.
  3. Gastos para el mantenimiento de la empresa: No procede.
  4. Gastos en representación: No procede.

  Total: $0.00.
- ✓ ***Gastos de distribución y venta*.**
  1. Participación en ferias o exposiciones: No procede.
  2. Gastos en transportación: No procede.
  3. Compra de materiales de propagandas: No procede.

  Total: $0.00.

Costos en Moneda Nacional:

- ✓ ***Costos Directos.***

  1. Salario del personal que laborará en el proyecto: $100.00.
  2. El 12% del total de gastos por salarios se dedica a la seguridad social: No procede.
  3. El 0.09% de salario total, por concepto de vacaciones a acumular: No procede.
  4. Gasto por consumo de energía eléctrica: $ 6.00.
  5. Gastos en llamadas telefónicas: No procede.
  6. Gastos administrativos: No procede.

- ✓ ***Costos Indirectos.***
    1. Como saber: $ 100.00.

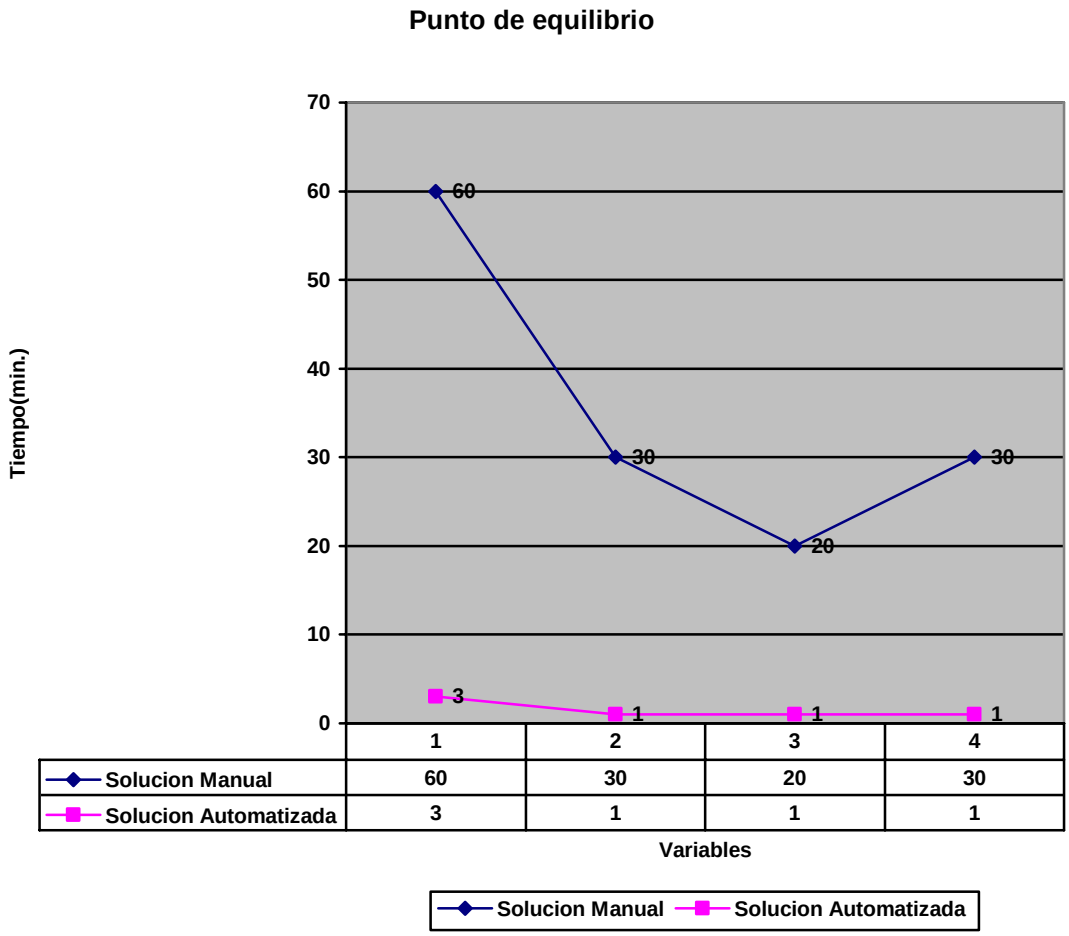
Total: $ 206.00.

Valores de la variable (Solución manual):

1. Determinar la cantidad de problemas que obligan al desarrollo del proyecto, determinar las posibles soluciones que se le darán a estos problemas. (60min)
2. Realizar una distribución de un número de involucrados a un proyecto a desarrollar. (30min)
3. Distribuir los involucrados relacionados con un proyecto a cada una de las actividades del mismo proyecto. (20min)
4. Generar el documento con los datos utilizados para la elaboración de una matriz de marco lógico. (30min)

Valores de la variable (Solución con el programa):

1. Recopilación de la información necesaria para iniciar el proceso de planificación y diseño del proyecto (proyecto a realizar, objetivos del proyecto, actividades por objetivo) (3min).
2. Generar reportes de las características de un proyecto o sus objetivos. (1min).
3. Relacionar un número de involucrados a un proyecto (1min).
4. Generación de la matriz de marco lógico según un proyecto definido. (1min)



5.4 Conclusiones

En este capítulo se realizó el estudio de factibilidad mediante La Metodología Costo Efectividad (Beneficio), se analizó los efectos económicos, así como se calculó el costo de ejecución del proyecto mediante la ficha de costo arrojando como resultado $ 40.00 CUC y $ 206.00 MN demostrándose la factibilidad del proyecto.

Conclusiones generales

Como resultado de la investigación se han podido lograr los objetivos propuestos y determinar los elementos expuestos en la hipótesis, para lo cual se logró desarrollar un software en el que se da cumplimiento a las especificidades propuestas, por lo que teniendo en cuenta estos elementos arribamos a las siguientes conclusiones:

Primera: La planificación y proyección de proyectos en la división Guantánamo se realizaba de forma manual, el control de dichos proyectos se archivaban con gran peligro de deterioro y perdida de los archivos en los que se plasmaba dicha planificación.

Segunda: Para dar solución a estos problemas se diseñó e implementó una aplicación que contiene una interfaz gráfica de usuario, para realizar la entrada de datos al sistema y mostrar datos generados.

- ✓ Se llevó a cabo un estudio de las principales metodologías, lenguajes y herramientas que se consideraron factibles para el desarrollo del sistema.
- ✓ Se realizó todo el proceso de desarrollo del software siguiendo las fases de la metodología XP, lo cual queda plasmado en el presente documento.

Tercera: Se pudo constatar con el personal especialista de la Empresa DESOFT S.A División Guantánamo que con el empleo de este software se agiliza todo el proceso de planificación, organización, análisis y control de los proyectos de desarrollo de software, arrojándoles beneficios en la calidad de los proyectos que desarrollan, así como una correcta aplicación de los costos de estos.

Recomendaciones

- Aplicar este software en la empresa DESOFT S.A. división Guantánamo para la organización, planificación, análisis y control de los proyectos de software.
- Extender las funcionalidades acordes a nuevos requisitos que surjan en la empresa DESOFT S.A.
- Realizar un estudio más profundo de este sistema en vista a perfeccionarlo en nuevas versiones del software.

Bibliografía

Barbone, Víctor A. González: XP: Extreme Programming.
http://iie.fing.edu.uy/~nacho/blandos/seminario/XProg1.html.

Lsi.us.es: PSISEXTREMA.pdf. [En línea] 20 febrero 2010.
http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf

Patricio Letelier, Ma. Carmen Penadés: Metodologías ágiles para el desarrollo de software: Extreme Programming (XP). [En línea] 22 febrero 2010.
http://www.willydev.net/descargas/masyxp.pdf.

Carlos Alberto Vigil Taquechel: Algunas Ideas Claves para la gestión de proyectos internacionales de las universidades cubanas.

Herman H. Grant: Logical Framework (Log frame) Methodology.

Eduardo Aldunate: Matriz de indicadores y gestión de programas.

J. J Gutiérrez, M. J Escalona, M. Mejías, J. Torres: Pruebas del sistema en programación extrema

Msc Alejandro Aguilera Sierra: Introducción a la programación extrema.

Wilder Escobedo D: Programación extrema.

Glosario de Términos

Software: Es la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo.

Software Libre: es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente, aunque conserve su carácter de libre, puede ser vendido comercialmente.

Programación Extrema(XP): Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo**.**

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

Metodologías de Desarrollo: Se define como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información.

Metodología Ágil: Constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de

desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración.

Metodología de marco lógico: es una herramienta analítica, para la planificación de la gestión de proyectos orientado por objetivos. Es utilizado con frecuencia por organismos de cooperación internacional.

Anexos

1. Ejemplo de un reporte con los elementos que conformaran la matriz de marco lógico y la estructura de la misma.

Nombre del Proyecto: *Capacitación*

Objetivos	Propósito	Fuente de Verificación Objetivo	Supuesto Objetivo	Indicador Objetivo	Actividades	Fuente de Verificación Actividad	Supuesto Actividad	Indicador Actividad
Realizar la capacitación en la empresa	Capacitar a los trabajadores	Desoft	a fines de este mes	Eficiencia				
					impartir clases de base de datos	yoangel	terminar la conferencia	Calidad

2. Ejemplo de cómo se relacionan los involucrados con los distintos proyectos que se han definidos para el sistema.

Proyecto que pertenece	Nombre	Apellidos
capacitación	Onésimo	Mustelier Castillo
	Delmis Wilfredo	Carbonel
	William	Hernández
	Luis Humberto	Batista
Fernando	Delmis Wilfredo	Carbonel
	Fernando	Samón Cunninghan
	Luis Humberto	Batista
realizar la tesis de grado	Yoangel	Hernández García
	William	Hernández

3. Reporte en el cual se da a conocer por cada objetivo de un proyecto definido su por ciento de realización, que se realiza utilizando los datos siguientes:
    - Objetivo a realizar el cálculo de porcentaje.
    - Cantidad total de actividades para este objetivo.
    - Cantidad de actividades cumplidas para el objetivo seleccionado.

Objetivos	Cantidad Actividades para este Objetivo	Actividades Cumplidas	Por ciento
Realizar la capacitación en la empresa	1	1	100