Universidad de las Ciencias Informáticas

Facultad 3

Título: “Análisis y diseño de las técnicas Análisis de Secuencia y Control de Términos pertenecientes al módulo Técnicas para la plataforma de Arquitectura de la Información ABAD”.

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

Autores: Diamelys Díaz Estrada.

Enrique Orta Sueiro.

Tutores: Ing. Yanicet Aveleira Rodríguez.

Ing. Eliober Cleger Despaigne.

Ciudad de la Habana, Cuba

Enero 2010.

No es sabio aquel que sabe donde está el tesoro, sino el que trabaja y lo saca.

Francisco de Quevedo y Villegas

DECLARACIÓN DE AUTORÍA

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

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

Diamelys Díaz Estrada

_____

Firma del Autor

Enrique Orta Sueiro

_____

Firma del Autor

Yanicet Aveleira Rodríguez

_____

Firma del Tutor

Eliober Cleger Despaigne

_____

Firma del Tutor

Agradecimientos

A nuestro tutor Ing. Eliober Cleger Despaigne por todo el apoyo y ayuda brindada, sin él este trabajo no hubiera sido posible, por su paciencia y dedicación en cada momento y a pesar de encontrarse cumpliendo con su deber en la República Bolivariana de Venezuela siempre está al tanto de nuestro trabajo.

A nuestra tutora Ing. Yanicet Aveleira Rodríguez por su apoyo, su valiosa colaboración y buena voluntad en las actividades, así como en sus observaciones críticas en el trabajo.

A la Ing. Diana Valdés González por su predisposición permanente e incondicional en aclarar nuestras dudas y por sus sugerencias durante la redacción de la Tesis, por su amistad.

A nuestras amistades que a lo largo de estos cinco años han compartido momentos de alegrías y tristezas, a los cuales consideramos como de nuestra familia, por su cariño, amistad y apoyo.

A nuestra familia sabiendo que jamás encontraremos la forma de agradecer su constante apoyo y confianza, sólo esperamos que comprendan que nuestros esfuerzos y logros han sido también suyos e inspirados en ustedes.

Diamelys y Enrique

Dedicatoria

A mis padres quienes me han heredado el tesoro más valioso que puede dársele a un hijo: amor. A quienes sin escatimar esfuerzo alguno, han sacrificado gran parte de su vida para formarme y educarme. A quienes la ilusión de su vida ha sido convertirme en persona de provecho. A quienes nunca podré pagar todos sus desvelos ni aún con las riquezas más grandes del mundo.

Por esto y más… Gracias.

Diamelys

A mis padres y hermanita por el cariño y apoyo moral que siempre he recibido de ustedes, que es para mí la mejor de las herencias y con el cual he logrado superar cada una de las pruebas que la vida ha puesto en mi camino. A ustedes les dedico este trabajo y toda mi vida, que nunca compensará el esfuerzo y el amor que me han dado.

Espero nunca defraudarlos...

Enrique

RESUMEN

El presente trabajo está enmarcado en el objetivo de realizar el análisis y diseño del módulo Técnicas de la Herramienta CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) para la Arquitectura de Información "ABAD", relacionado con el proyecto "ABAD" de la infraestructura productiva de la Universidad de las Ciencias Informáticas(UCI), con el cual se pretende, de forma amena e interactiva crear una Herramienta CASE, que apoye el proceso de Arquitectura de Información (AI). Este documento recoge los resultados del trabajo realizado en el módulo Técnicas, específicamente las técnicas Análisis de Secuencia y Control de Términos de las que no se tienen ningún tipo de documentación, ni artefactos que permitan la implementación de estas técnicas, se hace un estudio exhaustivo de los principales modelos, metodologías y estándares para el desarrollo de este tipo de software, así como de las tendencias y tecnologías actuales, permitiendo seleccionar las más adecuadas que apoyen la solución del problema y las herramientas de desarrollo a emplear en la producción del mismo. Se cumplió el objetivo propuesto a partir del empleo de la metodología RUP (Proceso Unificado de Desarrollo de Software), utilizando además BPMN (Business Process Modeling Notation, Notación para el Modelado de Procesos de Negocio) como extensión de UML (Unified Modeling Language, Lenguaje de Modelamiento Unificado) para el modelado del negocio, lográndose identificar las funcionalidades y el modo en que éstas deben ser implementadas para obtener el modelado que facilitará, viabilizará y apoyará el proceso de implementación y producción del software. Como conclusiones generales, se determina que los objetivos planteados han sido cumplidos de manera satisfactoria, y como complemento se proponen una serie de recomendaciones para posteriores mejoras.

ÍNDICE

INTRODUCCIÓN **1**

CAPÍTULO1: FUNDAMENTACIÓN TEÓRICA **5**

INTRODUCCIÓN 5

1.1 Arquitectura de Información *5*

1.2 Técnicas de Arquitectura de Información *6*

1.2.1 Análisis de Secuencia 7

1.2.2 Control de Términos 9

1.3 Técnicas matemáticas en el proceso de información *10*

1.3.1 WebSort 10

1.3.2 OptimalSort 10

1.3.3 SynCaps 10

1.3.4 UXSORT 11

1.4 Aplicaciones de Escritorio vs Aplicaciones Web *11*

1.5 Metodologías de desarrollo del software *12*

1.5.1 Extreme Programing (XP) 13

1.5.2 SCRUM 14

1.5.3 Rational Unified Process (RUP) 15

1.5.4 ¿Por qué usar RUP? 17

1.6 Business Process Modeling (BPM) *18*

1.7 Ingeniería de Requisitos. *19*

¿Qué es la Ingeniería de Requisitos (IR)? *19*

1.8 Lenguajes de Programación. *20*

1.8.1 Python 20

1.8.2 C++ 21

1.8.3 Java 21

1.8.4 ¿Por qué utilizar Java? 22

1.9 Patrones *23*

1.9.1 Patrones de diseño 23

Patrones GoF 24

1.10 Herramientas *25*
1.10.1 Rational Rose 25
1.10.2 Visual Paradigm para UML 26
1.10.3 ¿Por qué Visual Paradigm? 27
CONCLUSIONES 28

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA 30

INTRODUCCIÓN 30
2.1 Características del Sistema *30*
2.2 Modelos de procesos del Negocio utilizando BPM *30*
2.2.1 Modelo de Procesos del Negocio de la Técnica Análisis de Secuencia 30
2.2.2 Modelo de Procesos del Negocio de la Técnica Control de Términos 33
2.3 Requisitos Funcionales *34*
2.4 Requisitos no Funcionales. *38*
2.5 Actores del Sistema *39*
2.6 Estructuración del Diagrama de Casos de Uso del Sistema. *39*
2.6.1 Especificación textual de los casos de uso. 39
CONCLUSIONES 45

CAPÍTULO 3: ANÁLISIS Y DISEÑO DE LAS TÉCNICAS ANÁLISIS DE SECUENCIA Y CONTROL DE TÉRMINOS. 46

INTRODUCCIÓN 46
3.1 Análisis *46*
3.1.1 Diagramas de Clases del Análisis 46
3.1.2 Realización de casos de uso del análisis 47
3.2 Diseño *49*
3.2.1 Arquitectura definida para el sistema 49
3.2.2 Realización de Casos de Uso 50
CONCLUSIONES 52

CAPÍTULO 4: VALIDACIÓN 53

INTRODUCCIÓN 53
4.1 Proceso o fase inicial. Modelo de métricas *53*
4.1.1 Métrica de la calidad de especificación de los requisitos 54

4.1.2 Aceptación del cliente *56*

4.1.3 Métricas aplicadas en el Diagrama de Casos de Uso del Sistema 57

4.1.4 Métricas orientadas a clases. Tamaño de clase (TC) 64

CONCLUSIONES 65

CONCLUSIONES GENERALES 66

RECOMENDACIONES 67

BIBLIOGRAFÍA 68

INTRODUCCIÓN

La AI es una nueva disciplina encargada del estudio, análisis, organización y estructuración de la información en espacios de información, y de la selección y presentación de los datos en los sistemas de información interactivos. Uno de sus principales objetivos es que la asimilación de contenidos por parte del usuario sea eficiente y efectiva, y que la información pueda calificarse como accesible y usable. El resultado de la AI constituye uno de los requisitos no funcionales más importantes para el éxito de un proyecto de software.

Usualmente el trabajo de los Arquitectos de Información se concreta en la generación de un conjunto de materiales entregables en los que se plasma de forma efectiva la estructura del espacio de información, el diseño de la interacción del usuario con el sistema y el funcionamiento de la interfaz. Estos entregables tienen la finalidad de servir:

- A los clientes como resultado del proceso de diseño o reingeniería del software llevado a cabo durante la fase de análisis, definición y desarrollo del proyecto.
- A los diseñadores gráficos como material de base para la producción de maquetas gráficas sobre las que se basarán los maquetadores.
- A los ingenieros informáticos para desarrollar la programación del software.

En los proyectos de software se usan herramientas CASE, para la generación de los artefactos que la metodología seleccionada propone. Las mismas son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. (1)

Desde que se crearon en (1984) hasta la actualidad, las CASE cuentan con una credibilidad y exactitud de reconocimiento universal, siendo usadas por cualquier desarrollador y / o programador que busca un resultado óptimo y eficiente, pero sobre todo que busca esa minuciosidad necesaria de los procesos y entre los procesos.

La UCI hoy cuenta con más de 150 proyectos productivos entre los que se encuentran los proyectos de exportación que se desarrollan principalmente con la República Bolivariana de Venezuela, proyectos nacionales que se desarrollan con más del 80% de los ministerios cubanos y proyectos de informatización internos de nuestra universidad, de estos proyectos hasta el momento de la experiencia obtenida se puede concluir que la aplicación de la arquitectura de información así como su comercialización se ha visto afectada por algunas limitaciones que a continuación se describen:

- Se utilizan varias herramientas informáticas para el proceso de AI por ejemplo la herramienta Microsoft Office Word como procesador de texto, MindManager para la realización de mapas conceptuales, el Microsoft Office Photoshop para la realización de prototipos de interfaz gráfica, entre otras, lo que hace que el trabajo sea más engorroso a la hora de trabajar por la diversidad de herramientas que hay que utilizar.
- Dentro del proceso; la realización de técnicas tampoco cuenta con una herramienta que integre las mismas, sino que se encuentran independiente una de otras.
- La AI es un servicio altamente comercializable, sin embargo, en la UCI este proceso se torna complejo, teniendo en cuenta que las herramientas existentes son mayormente propietarias y debe analizarse el problema asociado a las licencias.

El Centro de Informatización de la Infraestructura productiva tiene entre sus compromisos realizar una Herramienta CASE que sirva de apoyo a la realización del proceso de AI en los proyectos productivos de la UCI y elimine los defectos que las limitaciones antes mencionadas incorporan al proceso. Para cumplir el compromiso se creó el proyecto ABAD, el mismo consta de cuatro módulos, uno de ellos tiene como objetivo informatizar las técnicas que los Arquitectos de Información utilizan. Una de las tres técnicas que inicialmente se quieren incorporar a la herramienta se encuentra en la etapa de desarrollo pero las restantes no cuentan con los artefactos necesarios para que los desarrolladores continúen la implementación del módulo Técnicas.

Analizando la anterior problemática, se ha planteado el siguiente **problema científico:** La ausencia de artefactos requeridos para implementar las técnicas Análisis de Secuencia y Control de Términos impide que los desarrolladores completen la implementación en el módulo Técnicas de la Herramienta CASE para la Arquitectura de Información "ABAD". Por lo tanto, el **objeto de estudio** de la investigación sería**:** La Ingeniería de Software, centrando la atención en el **campo de acción:** Flujos de trabajo Requisitos,

Análisis y Diseño. Con el propósito de dar solución al problema se precisa como **objetivo general:** Obtener los artefactos necesarios para la posterior implementación de las técnicas análisis de secuencia y control de términos. **Hipótesis:** Si se obtienen los artefactos de los flujos de trabajo requisitos, análisis y diseño para las técnicas Análisis de Secuencia y Control de Términos se le facilitará el trabajo a los desarrolladores para llevar a cabo la implementación del módulo.

Para alcanzar dicho objetivo se trazaron las siguientes **tareas de la investigación:**

1. Realizar el estudio del estado del arte de las diferentes técnicas y herramientas utilizadas por los arquitectos de información en los proyectos informáticos, así como la metodología y herramienta de desarrollo definiéndose los patrones de caso de uso y de diseño para el desarrollo del módulo.
2. Realizar el proceso de Ingeniería de Requisitos para las técnicas de Análisis de Secuencia y Control de Términos.
3. Realizar el análisis y diseño de las técnicas Análisis de Secuencia y Control de Términos.
4. Identificar las métricas que serán aplicadas para la validación del diseño propuesto.

Métodos y técnicas utilizadas en la investigación:

Métodos Teóricos:

Histórico – Lógico:

Para analizar la trayectoria y evolución de la metodología de desarrollo de Software y demás herramientas que se utilizarán durante el trabajo.

Métodos Empíricos:

Entrevista:

Para recopilar las características más importantes de los procesos del negocio al cual responde el sistema, obtenidos directamente de los usuarios.

Observación:

Con el objetivo de analizar y captar deficiencias en el proceso de análisis.

Revisión bibliográfica:

Para realizar un estudio de un conjunto de fuentes de información referidas al tema, así como libros, artículos, revistas, publicaciones, boletines y una especie de materiales escritos y digitalizados, que son

de gran utilidad para documentar la base teórica del trabajo a desarrollar, debidamente referenciadas para futuras consultas sobre el tema.

Estructura de la tesis.

El presente trabajo, estructurado en cuatro capítulos, resume la siguiente información:

Capítulo 1. Se realizará el estado del arte referente a las técnicas usadas por los arquitectos de información y características de las herramientas CASE. Metodologías, herramientas y patrones utilizados, donde se analizan las metodologías y herramientas más usadas actualmente en la producción de software y se fundamentará el motivo de la selección.

Capítulo 2. Se aplicará la ingeniería de requisitos desarrollándose las tres primeras etapas: la elicitación o captura de requisitos para poder guiar el desarrollo del software hacia el sistema correcto; el análisis de requisitos para determinar el dominio de la aplicación, el desempeño del sistema, así como las restricciones que el sistema debe poseer, entre otras cosas; la especificación de los requisitos para determinar lo que el sistema debe realizar; además de las descripciones de los casos de uso para una mayor comprensión de las acciones en cada uno y el esclarecimiento del diseño del sistema a construir. Se obtienen los artefactos generados a partir de los flujos de trabajo de negocio y requisitos.

Capítulo 3. Se continúa con el flujo de Análisis y Diseño del sistema donde se obtienen: los diagramas de clase del análisis y los diagramas de colaboración correspondientes, el modelo del diseño con los artefactos: diagramas de clases del diseño por subsistema de diseño y los diagramas de secuencia.

Capítulo 4. Se realiza la validación de los requisitos y del diseño propuesto mediante la aplicación de métricas y técnicas estudiadas.

CAPÍTULO1: Fundamentación Teórica

Introducción

En el presente capítulo se recogen los principales conceptos relacionados con el tema de la AI y herramientas CASE, explicándose la tecnología, los componentes y las estructuras de estas herramientas. Además, se exponen las características de las principales técnicas de AI y herramientas que son utilizadas para el desarrollo del módulo. También se aborda acerca de las diferentes metodologías para el proceso de desarrollo de software, los patrones a utilizar en el desarrollo del módulo para llegar finalmente a la conclusión de qué se debe utilizar en la solución propuesta.

1.1 Arquitectura de Información

“Una arquitectura de información es un diseño o plano para modelar los requisitos globales de información de una empresa. Proveyendo una forma de representar las necesidades de información de una organización, relacionando a ellas los procesos específicos de negocio y documentando sus interrelaciones. Este mapa de procesos de información es usado entonces para guiar el desarrollo de aplicaciones y para facilitar integrar y compartir datos.”(3)

La organización de la información no es más que el proceso donde se dispone y ordenan los elementos secuencialmente que integran el contenido de un sitio web, software o multimedia. En este proceso, se consideran las distintas características de los sistemas de clasificación y ordenamiento como son la ambigüedad, la heterogeneidad y la homogeneidad. Además, se seleccionan los esquemas de organización de la información y las estructuras de organización de la información que se utilizarán. (4)

En la AI se trata de organizar la información de forma tal que los usuarios puedan encontrar las respuestas correctas a sus interrogantes. Su objetivo radica en la creación de sistemas de organización de la información y etiquetados términos que designan o describen una entidad que realmente posean un significado para los usuarios.(4)

La AI no es más que arte y ciencia, a partir de necesidades, características de usuarios y su entorno trata de definir las estructuras organizacionales de la información y los métodos con que interactuarán con ella. Esto lo logra interactuando con los usuarios, estudiando el contexto, organizando y representando la información. Lo hace para lograr una mayor calidad del producto de información (mejor búsqueda y recuperación, usabilidad, experiencia, etc.). Con lo que trabaja es con la información (recursos de

información, productos de información, etc.). Como resultado se generan: Informes, diagramas, mapas, tablas. Se puede desarrollar la AI de forma digital (software, sitios web, intranet, sistemas de información, etc.). Se utiliza fundamentalmente para el diseño de interacción, el diseño de información, el diseño gráfico y la programación.(5)

1.2 Técnicas de Arquitectura de Información

Técnicas de interacción con el usuario

Son técnicas que permiten obtener información relacionada con los usuarios del producto final. Esta información es la base para lograr un diseño centrado en el usuario, sobre la que se sustentan las posteriores etapas de producción. Dentro de este tipo de técnicas se encuentran: reunión, entrevista y encuesta, diseño de escenarios y el diseño participativo.(6)

Técnicas de interacción con el contexto

Son técnicas que buscan información de productos similares o productos que le hagan lo mismo que el que se está realizando. El objetivo principal de esta técnica es conocer qué cualidades tienen los productos similares o de la competencia, para poderlos mejorar y superar; o qué dificultades tienen estos productos revisados, para no repetirlas en el que se está desarrollando, dándole solución y obteniendo una ventaja competitiva. Dentro de las técnicas de interacción con el contexto se encuentran: evaluación de productos similares y análisis de la competencia. (6)

Técnicas matemáticas (coocurrencia)

Esta técnica consiste en la aplicación del análisis de coocurrencia para cuantificar resultados y hacer precisa la toma de decisiones. Con la aplicación de estas técnicas se logran definir grupos y crear secuencias que se correspondan con el modelo mental de los usuarios. Dentro de las técnicas matemáticas se encuentran: Organización de tarjetas (cardsorting) y Análisis de Secuencia.(6)

Técnicas de representación de información

Son las técnicas que contribuyen a concretar las propuestas de diseño establecidas por los productores de manera abstracta. Consiste en la creación de modelos y prototipos de lo que debe ser el producto final. Los modelos facilitan la retroalimentación de los criterios y necesidades de los usuarios en cuanto a las soluciones de diseño del producto. Este grupo de técnicas se realizan a partir de la información que se

obtiene de las técnicas anteriores. Dentro de las técnicas de representación de información de encuentran: bocetado, representación de etiquetas y prototipado. (6)
Las técnicas que se utilizarán en el proyecto aparecen descritas a continuación:

1.2.1 Análisis de Secuencia

La AI tiene como función fundamental la organización y representación de términos en el etiquetado de productos electrónicos, principalmente sitios web o multimedias interactivas. Para realizar esta actividad es necesario hacer un análisis de información a partir de las necesidades y requisitos de los usuarios potenciales del producto. Dentro de esta técnica lo principal es hacer una organización secuencial, es decir, darle orden a las etiquetas que se seleccionen de forma tal que sean representadas de la mejor forma posible las estructuras mentales de los usuarios finales. Es por esto que la organización secuencial de los términos en una barra de navegación o menú desplegable es de gran importancia para el posterior desempeño satisfactorio del producto. (7)

Generalmente existe el dilema de no saber cuál será el orden de los términos. Existen varias formas de organizarlos secuencialmente como son: la numérica, la alfabética o la cronológica. Cuando los términos no son susceptibles a estos tipos de organización se puede ubicar de acuerdo con el criterio personal de arquitectos de información, lo que no se podría estar seguro de que los usuarios del sitio web o la multimedia interactiva se identifiquen con la selección individual. También se podría organizar secuencialmente los términos siguiendo una lógica funcional, temporal o estructural. Por ejemplo:

- Una organización funcional sería: agua, azúcar, limón, limonada (siguiendo un principio lógico de funcionamiento).
- Una organización temporal sería: bisabuelos, abuelos, padres, hijos (siguiendo un principio temporal de secuencia).
- Una organización estructural sería: gerente, subgerente, jefe de grupo, guardia de seguridad (siguiendo un principio estructural de una empresa).

Puede haber muchas formas de organizar secuencialmente siguiendo distintos principios basados en la lógica, pero en ocasiones estos principios no son suficientes. Con el objetivo de obtener una información más cercana posible a la necesidad del usuario final y definir la secuencia de acorde con esto, se propone complementar los métodos que existen, con lo que se llama un Análisis de Secuencia. (7)

La técnica de Análisis de Secuencia consiste en hacer un conjunto de pruebas a los usuarios potenciales del producto y un análisis cualitativo y cuantitativo de los resultados que arrojen las pruebas para de esta forma poder definir la secuencia de las etiquetas en el producto electrónico. Esta técnica se asemeja mucho al método de organización de tarjetas (cardsorting), la diferencia es la finalidad del análisis. En el cardsorting el objetivo es definir las distintas agrupaciones de las etiquetas y en el Análisis de Secuencia se define el orden que van a tener las mismas. (7)

Una vez obtenidos los resultados de los usuarios, lo que se realiza es un análisis de coocurrencia. El análisis de coocurrencia consiste en el estudio de la aparición conjunta de dos o más etiquetas y/o grupo de etiquetas. Es decir, se analiza cuántas veces se repiten los distintos criterios de secuencialización que establecen los usuarios, según el método que se realice en cuestión (cardsorting o análisis de secuencias). Por ejemplo, en el cardsorting, se analiza la cantidad de veces que se repiten los distintos tipos de agrupaciones de tarjetas que hacen los usuarios; lo cual conduce a un análisis más certero en cuanto al agrupamiento de etiquetas; mientras que en el Análisis de Secuencia se analizan la cantidad de veces que se repiten los tipos de secuencia. Es por ello que se recomienda que se complementen ambos métodos. En este caso, se explicará el método del Análisis de Secuencias, como una opción ventajosa a la hora de diseñar el ordenamiento de las etiquetas. (7)

Los pasos que se proponen para el Análisis de Secuencia son los siguientes:

1. Definición de los términos (si son de una barra de navegación, un menú desplegable o una lista pequeña de términos). En este paso se aconsejan basarse en las técnicas de entrevista y encuesta del diseño centrado en el usuario. Es importante solicitarle a los usuarios seleccionados en la muestra que expliciten los términos que consideran ellos que deben usarse en la interfaz como etiquetas. Es útil también, usar la técnica de la revisión bibliográfica en este paso.
2. Realización de tarjetas con cada término acordado durante la realización de las técnicas anteriores; cada tarjeta debe tener escrito el término con la posibilidad de que se le pueda escribir otro término como sugerencia. Cada tarjeta debe tener escrito un número en la parte posterior, con el objetivo de facilitarle el ordenamiento al arquitecto de información y poderlo tabular para realizar el análisis cuantitativo.
3. Entrega del grupo de tarjetas a una muestra de usuarios potenciales y solicitarle a los mismos que las organicen consecutivamente (1, 2, 3…) según su criterio.

4. Realización del análisis cualitativo de la prueba. En el análisis cualitativo se observan qué términos ofrecieron dificultad, cuál no se comprendió, cuál es el primero y porqué, qué criterio de organización usó el usuario, qué otro término propone algún usuario, cuál término resulta ambiguo o establece polisemia, etc.
5. Recogida de los resultados y análisis cuantitativo (se tabulan los resultados y se realiza el análisis de coocurrencia, luego se grafica).
6. Conclusiones (se define el orden secuencial, basándose en los análisis cualitativos y cuantitativos, y los principios lógicos de ordenamiento). (6)

1.2.2 Control de Términos

La técnica de control de términos, consiste al igual que la técnica de análisis de secuencia en una serie de pruebas a usuarios potenciales del producto, y el posterior análisis cualitativo y cuantitativo de esos resultados; para ayudar a definir cuál es el término preferente mientras que análisis de secuencia realiza un ordenamiento de etiquetas para definir la secuencia.

Los pasos que se proponen para el control de términos son los siguientes:

1. Se definen las distintas variantes o nombres que se le pueden otorgar al término.

2. Se le entrega a los usuarios un grupo de tarjetas con las distintas variantes que contienen el término y un número asociado para que éstos escojan el término de su preferencia (Término dominante), el cual significa que es el más aceptado entre todos y se le da la oportunidad de hacer otra propuesta de término en caso que piense que pueda haber otro más acorde.

3. Se realiza el conteo cuantitativo de los términos seleccionados por los usuarios para determinar cuál será el término preferente.

4. Conclusiones: se define cual es el término dominante.

Por lo que se necesita conocer a través de una técnica matemática que cuantifique e ilustre cuál es el término preferido por la mayoría de los usuarios, de manera que se pueda usar la técnica cuando se le aplique a un número mayor de usuarios y con un número mayor de términos.

1.3 Técnicas matemáticas en el proceso de información

1.3.1 WebSort

WebSort permite realizar a distancia, en línea la clasificación de tarjetas. Crear un estudio, enviar un vínculo a los participantes y analizar los resultados, todos a través de una sencilla interfaz basada en web.

WebSort es una plataforma que permite recolectar información de los visitantes de forma ordenada y luego, presentarla a los responsables para que analicen los datos.

Esta técnica puede ser aplicada en muchos sectores, por ejemplo, para decidir qué áreas poner en un sitio, también se puede aplicar a la usabilidad, ya que se pueden obtener datos sobre qué tan fácil es navegar por la Web, según los mismos visitantes. En resumen, es un servicio para recolectar opiniones sobre un sitio Web, opiniones importantes que deben considerarse.

También se pueden compartir los resultados de cada estudio, por si es que se trabaja en un grupo. Gráficas, reportes, comentarios, todo se puede compartir. WebSort ofrece un estudio gratis para todo aquel que se registre, pero también existen planes que se ofrecen de acuerdo a las necesidades de cada cliente. (7)

1.3.2 OptimalSort

OptimalSort permite ejecutar las clases en línea de tarjetas con un mínimo de alboroto. La clasificación de las tarjetas es un método confiable para entender cómo piensan las personas acerca de los contenidos. Usando OptimalSort se puede llegar a los participantes en cualquier parte del mundo, salvo las horas de trabajo manual y obtener una visión inmediata de los resultados. (8)

1.3.3 SynCaps

SynCaps edición V2 proporciona análisis visual más sofisticado y características para la manipulación de datos ordenados, como la fusión de elementos o grupos. Ambos proporcionan un tradicional "dendrograma", un gráfico en forma de árbol que muestra la fuerza relativa de las relaciones entre los elementos en base a la frecuencia con que se presentaron juntas en grupos. (9)

1.3.4 UXSORT

UXSORT es una aplicación de clasificación de las tarjetas que permite a la usabilidad / experiencia de usuarios profesionales para planificar las actividades de tarjeta, clasificar, gestionar usuarios y tarjetas, recoger datos del usuario clasificación de las tarjetas, analizar datos y crear un informe final.

UXSORT es desarrollado por un investigador de la experiencia del usuario en el tiempo libre para ayudar a los equipos internos para llevar a cabo las fases de procesamiento de tarjeta rápida y ver los resultados inmediatamente. Está diseñado particularmente para la clasificación de la tarjeta en un gran número de elementos (hasta 1000 tarjetas) en un nivel de estructura jerárquica-multi.

UXSORT (v1.7, v1.6, v1.5, v1.0) es gratuito. El uso de uxsort debe cumplir con el acuerdo de licencia GNU.(10)

1.4 Aplicaciones de Escritorio vs Aplicaciones Web

Los sistemas informáticos se pueden clasificar en aplicaciones de escritorio, aplicaciones web, aplicaciones por consola, entre otras muchas categorías. Estas responden a la forma en la cual un software se distribuye, a la forma de distribución de los datos en el mismo, a la forma de instalación, y forma de procesamiento de los datos. Las dos categorías que más corresponden con los propósitos del presente trabajo son las aplicaciones de escritorio y las aplicaciones Web.

“Una aplicación web consiste en una página web con contenido dinámico, es decir, que cambia su contenido según la interacción de cada usuario que la visita.”(11) No es necesaria la instalación por parte de los usuarios del software en sus computadoras sino tener instalado un navegador web para visualizarlo y acceso a una red que permita la conexión a donde la aplicación de se encuentre alojada.

Una aplicación de escritorio consiste en un software instalado directamente en las computadoras de sus usuarios (no es necesario el navegador web ni el acceso a una red).

Los usuarios finales de una aplicación informática a los cuales se les aplicarán las técnicas de Control de Términos y Análisis de Secuencia no siempre pudieran contar en su institución con una red de computadoras, por lo que un sistema que informatice las técnicas a través de una aplicación web no

tendría relevancia. Además, hay que tener en cuenta que la plataforma ABAD será una aplicación de escritorio.

Por lo que se propone desarrollar el sistema como una aplicación de escritorio debido a las características del entorno en el cual se aplica el sistema.

1.5 Metodologías de desarrollo del software

¿Qué es una metodología de desarrollo de software?

Las metodologías de desarrollo de software abarcan todo el ciclo de vida del software, y se definen como un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software(11), adoptando la misma se obtiene un producto software más predecible y permite ciertas características deseables como:

- Existencias de reglas bien definidas.
- Verificaciones intermedias.
- Planificación y control.
- Comunicación efectiva.
- Utilización sobre un abanico amplio de proyectos.
- Fácil formación.
- Herramientas CASE.
- Actividades que mejoren el proceso de desarrollo.
- Soporte al mantenimiento.
- Soporte de la reutilización de software.

De acuerdo con su definición de metodología, los procedimientos detallan consejos para elaborar una actividad; las técnicas serían la forma de ejecutar un procedimiento para obtener un resultado determinado; las herramientas de software son las que hacen posible automatizar el proceso de desarrollo del software y la documentación es la que identifica el software que se está desarrollando.(12)

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, lo que trae consigo son clientes y desarrolladores insatisfechos con el resultado obtenido. Sin embargo, muchas veces no se toma en cuenta el utilizar una metodología adecuada, sobre todo cuando se trata de proyectos pequeños de dos o tres meses. Lo que se hace con este tipo de proyectos es separar rápidamente el aplicativo en procesos, cada proceso en funciones, y por cada función determinar un tiempo aproximado de desarrollo(13). Cuando los proyectos que se van a desarrollar son de mayor envergadura, ahí sí toma sentido el basarse en una metodología de desarrollo, y empezar a buscar cual sería la más apropiada para este caso. Lo cierto es que muchas veces no se encuentra la más adecuada y se termina haciendo o diseñando una metodología propia, algo que por supuesto no está mal, siempre y cuando cumpla con el objetivo. (14)

Para dar una idea de qué metodología se puede utilizar en el desarrollo del sistema y cuál de éstas es más factible de utilizar se hace mención a tres.

1.5.1 Extreme Programing (XP)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada para proyectos de corto plazo, un pequeño equipo y cuyo plazo de entrega es el mínimo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.(15)

Características de XP, la metodología se basa en:

Pruebas Unitarias:

Se basa en las pruebas realizadas a los principales procesos, de tal manera que se adelante en algo hacia el futuro y que se puedan hacer pruebas de las fallas que pudieran ocurrir.

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.

¿Qué es lo que propone XP?

El manejo del cambio se convierte en parte fundamental del proceso. El costo del cambio no depende de la fase o etapa, no introduce funcionalidades antes que sean necesarias. El cliente o el usuario se convierten en miembro del equipo, derechos al cliente. Saber el estado real y el progreso del proyecto para añadir, cambiar o quitar requisitos en cualquier momento. Obtener lo máximo de cada semana de trabajo. Decidir cómo se implementan los procesos para crear el sistema con la mejor calidad posible. Pedir al cliente en cualquier momento aclaraciones de los requisitos para poder estimar el esfuerzo para implementar el sistema y empieza en pequeño y añade funcionalidad con retroalimentación continua.(15)

1.5.2 SCRUM

Es una metodología ágil de gestión de proyectos cuyo objetivo primordial es elevar al máximo la productividad de un equipo. Sólo abarca prácticas de gestión sin entrar en las prácticas de desarrollo como puede hacer XP. Delega completamente en el equipo la responsabilidad de decidir la mejor manera de trabajar para ser lo más productivo posible. SCRUM se basa en cierto “caos controlado”, el proceso del ciclo de desarrollo de SCRUM parte del conocimiento de que el proceso de desarrollo fundamental del producto está totalmente indefinido, es incluso caótico, pero establece ciertos mecanismos para controlar esta indeterminación, manipular lo impredecible y controlar la flexibilidad.(16)

En la metodología SCRUM se establecen tres fases: pre-aplicación, aplicación y post-aplicación:

En la pre–aplicación se definen y/o revisan las funcionalidades que ha de tener el sistema, haciendo la lista de funcionalidades que cuando estén completadas se podrá decir que el desarrollo del sistema está completo. Dicha lista recibe el nombre de Release Back Log. Una vez hecho esto se elige un subconjunto de estas funcionalidades con el que se va a trabajar el próximo mes, llamado Sprint Back Log y entonces comienza la siguiente fase.

En la aplicación es donde se desarrolla el Sprint, las reglas de esta fase son sencillas, se distribuyen las tareas para cada miembro del equipo, se trabaja duro y se intenta conseguir el objetivo. Todos los

miembros del equipo han de participar en una reunión diaria que en ningún caso deberá exceder los 30 minutos, llamada Sprint-meeting.

En esta reunión cada desarrollador debe dar respuesta a tres preguntas:

¿Qué hizo desde la última reunión?

¿Qué dificultades se están teniendo en el desarrollo de la tarea?

¿Qué va a hacer hasta la próxima reunión diaria?

En la post-aplicación se evalúa la entrega de funcionalidades del Sprint, se ven las tareas pendientes, se evalúa el progreso del proyecto y se redefine el tiempo de entrega del mismo si fuera necesario. En este paso se comparan las funcionalidades actuales con el Release Back Log y en el caso de no cumplirse se vuelve a la fase de pre-juego para realizar una nueva iteración. Si por el contrario, sí se cumplen entonces se pasa a la creación de la versión final y a la creación de la documentación pertinente.(16)

1.5.3 Rational Unified Process (RUP)

La metodología RUP, nombrada así por sus siglas en inglés Rational Unified Process, la cual divide en cuatro fases el desarrollo del software(14):

Inicio:

El objetivo en esta etapa es determinar la visión del proyecto, describir el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

Elaboración:

En esta etapa el objetivo es determinar la arquitectura óptima, desarrollar el plan del proyecto y eliminar los mayores riesgos.

Construcción:

En esta etapa el objetivo es llegar a obtener la capacidad operacional inicial del producto de forma incremental a través de las sucesivas iteraciones.

Transición:

El objetivo es llegar a poner en manos de los usuarios el producto final desarrollando nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del mismo y en general tareas relacionadas con el ajuste, configuración, instalación del producto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.(14)(17)

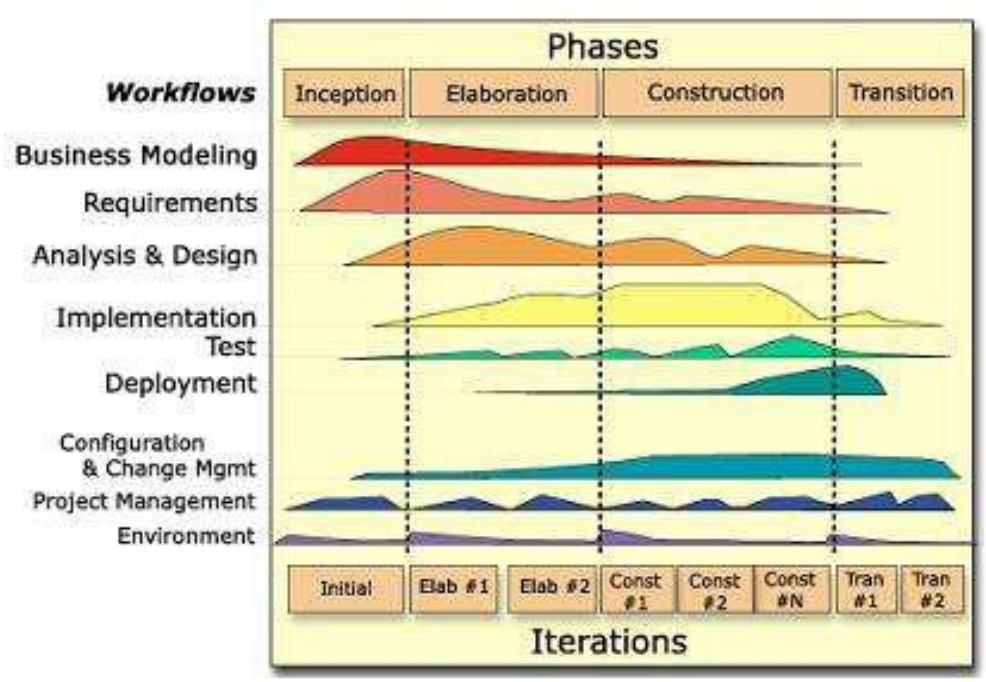


Figura 1: Fases e Iteraciones de la Metodología RUP

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierta luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Los elementos del RUP son:

Actividades:

Son los procesos que se llegan a determinar en cada iteración.

Trabajadores:

Vienen a ser las personas o entes involucrados en cada proceso.

Artefactos:

Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.(14)(17)

1.5.4 ¿Por qué usar RUP?

La metodología de desarrollo a utilizar es RUP debido a que:

- Aplica varias de las mejores prácticas en el desarrollo moderno de software en una forma que se adapta a un amplio rango de proyectos y de organizaciones.
- Reconoce que las necesidades del usuario y sus requisitos no se pueden definir completamente al principio.
- Permite evaluar tempranamente los riesgos en lugar de descubrir problemas en la integración final del sistema.
- Distribuye la carga de trabajo a lo largo del tiempo del proyecto ya que todas las disciplinas colaboran en cada iteración.
- Facilita la reutilización del código teniendo en cuenta que se realizan revisiones en las primeras iteraciones lo cual además permite que se aprecien oportunidades de mejoras en el diseño.
- Provee a cada miembro del equipo un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas del desarrollo de software.

Esta metodología permite que todos los integrantes de un equipo de trabajo, conozcan y compartan el proceso de desarrollo, una base de conocimientos y los distintos modelos de cómo desarrollar el software utilizando un lenguaje de modelado común: UML puede ser adoptado y extendido para satisfacer las necesidades de la organización que lo utilice seleccionando las fases e iteraciones, los flujos de trabajo y

disciplinas que se van a recorrer y los entregables o productos (artefactos) que se van a construir. Además de que se necesita tener documentación archivada para las posteriores versiones de la herramienta y RUP permite que se haga gran documentación sobre el proyecto al que se le aplica esta metodología.

1.6 Business Process Modeling (BPM)

En el presente trabajo se utilizó BPM para modelar los procesos de negocio, continuando luego con el flujo de trabajo requisitos propuesto por RUP.BPMN es un lenguaje de modelado para modelar conceptualmente procesos de negocio. Proporciona la capacidad de entender y definir dichos procesos, ya sean internos o externos, a través de diagramas de procesos de negocio. Se diseñó con el objetivo de facilitar la comprensión por parte de todos los implicados (expertos TIC, analistas de negocio, directivos, etc.) que participan en el proceso. El modelado de procesos de negocio suele empezar capturando actividades de alto nivel para luego ir bajando de nivel de detalle dentro de diferentes diagramas. Puede haber múltiples niveles de diagramas, dependiendo de la metodología usada para desarrollar los modelos. De todas formas, BPM es independiente de cualquier metodología.BPM define un Business Process Diagram (BPD), que se basa en una técnica de grafos de flujo para crear modelos gráficos de operaciones de procesos de negocio. Un modelo de procesos de negocio, es una red de objetos gráficos, que son actividades (trabajo) y controles de flujo que definen su orden de rendimiento.

Luego de estudiar los diferentes lenguajes de modelado se seleccionó BPMN para modelar los procesos de negocio y UML para el resto de los procesos.

BPM es elegido debido a la facilidad con que se describen las áreas de procesos y los elementos que en ellas interactúan para alcanzar el resultado esperado por los involucrados, ya que UML no está diseñado para modelar procesos del negocio y su enfoque orientado a objeto contradice el enfoque orientado al negocio, además de estar pensado para un público predominantemente técnico. BPM considera un único diagrama para representar los procesos de negocio, que está basado en la técnica de diagramas de flujo y adaptado para graficar las operaciones de los procesos de la organización. Está compuesto además por un conjunto de elementos gráficos que facilitan que tanto los miembros del equipo de desarrollo como el cliente entiendan con claridad sus diagramas. Constituye un puente para la separación que existe entre el diseño del proceso de negocio y la comunicación de requisitos de negocio.

1.7 Ingeniería de Requisitos (IR)

La parte más difícil en la construcción de sistemas software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan dificultosa como establecer los requisitos técnicos detallados, incluyendo todas las interfaces con humanos, máquinas y otros sistemas software. Ninguna otra parte del trabajo puede perjudicar tanto el resultado final si es realizada en forma errónea. Ninguna otra parte es tan dificultosa de rectificar posteriormente". (28)

¿Qué es la Ingeniería de Requisitos?

La Ingeniería de Requisitos se define como: " un conjunto de actividades en las cuales, utilizando técnicas y herramientas, se analiza un problema y se concluye con la especificación de una solución (a veces más de una)." (29)Entonces, "Ingeniería de Requisitos" se utiliza para definir todas las actividades involucradas en el descubrimiento, documentación y mantenimiento de los requisitos para un producto determinado. El uso del término "ingeniería" implica que se deben utilizar técnicas sistemáticas y repetibles para asegurar que los requisitos del sistema estén completos, sean consistentes y relevantes. La ingeniería de requisitos ha devenido, en la actualidad, en práctica ineludible para garantizar la calidad de productos informáticos, promoviéndose como disciplina clave en la ingeniería de software. Tradicionalmente el cliente se consideraba el máximo responsable de la definición de cada uno de los requisitos, y esta actividad se entendía como una parte borrosa del ciclo de vida del software. Desde mediados de los años 70 va cobrando una especial importancia, y hoy se considera una etapa clave en el desarrollo software, asumiéndose además que la elaboración de requisitos es una responsabilidad compartida entre clientes, usuarios, y equipo de desarrollo, teniendo como colofón la definición de la disciplina de Ingeniería de Requisitos. Y la IR no es más que el proceso sistemático de desarrollar requisitos mediante un proceso iterativo y cooperativo de analizar el problema, documentar las observaciones resultantes en varios formatos de representación y comprobar la precisión del conocimiento obtenido(30), contando con cuatro etapas fundamentales:

- Elicitación de Requisitos: se refiere a la captura y descubrimiento de los requisitos, en esta etapa se llevaron a cabo técnicas para capturar los requisitos como: entrevistas(son un método común, las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada, esta técnica permite que el equipo de trabajo se acerque al problema de una forma natural), arqueología de documentos(con la aplicación de este método se tratan de determinar posibles requisitos sobre la base de inspeccionar la documentación utilizada por la

empresa) y tormenta de ideas(este método se usa para generar ideas, la intención en su aplicación es de generar la mayor cantidad de requisitos para el sistema)**.**

- Análisis de Requisitos: consiste en detectar defectos y problemas entre requisitos, delimitar las fronteras del sistema y su interacción con el entorno.
- Especificación de Requisitos: se refiere a la documentación de cada uno de los requisitos capturados.
- Validación de Requisitos: consiste en demostrar que los requisitos definidos en el sistema son los que realmente quiere el cliente; además revisa que no se haya omitido ninguno, que no sean ambiguos, inconsistentes o redundantes.

1.8 Lenguajes de Programación

1.8.1 Python

“Python es un lenguaje interpretado que forma parte de un proyecto de software libre. Al no ser compilado como muchos otros lenguajes, su código no produce un ejecutable, sino que existe un intérprete que traducirá el lenguaje Python a código máquina.” (23)

Una vez que se desee cambiar de plataforma de desarrollo no habrá que cambiar el código sino el intérprete. El ser interpretado trae como consecuencia que la velocidad de ejecución frente a los lenguajes compilados es menor. Es orientado a objetos, se utiliza para construir cualquier tipo de aplicaciones.

Ha incorporado gran cantidad de librerías. Dispone de varias funciones para el tratamiento de las cadenas de texto, números, archivos. Además, se pueden importar librerías en los programas para tratar temas específicos como la programación de ventanas. Por ser un lenguaje de alto nivel y al tratarse de un lenguaje interpretado, el rendimiento no es óptimo.

Con Python no es necesario crear tantas líneas de código para la creación de programas como en otros lenguajes, por lo que la velocidad de desarrollo es mayor. Para la creación de aplicaciones de escritorio utilizando Python es necesario integrarlo con algunas librerías gráficas, como por ejemplo las que proporciona Qt (Biblioteca multiplataforma, creada para desarrollar interfaces gráficas de usuarios).

1.8.2 C++

C++ es un lenguaje imperativo orientado a objetos, que al igual que C (lenguaje predecesor), su ejecución sigue muy ligada al hardware subyacente manteniendo una considerable potencia para la programación a bajo nivel, aunque también permite un estilo de programación con un alto nivel de abstracción. Permite reutilizar código escrito en C fomentando la transición de los códigos escritos en este lenguaje a su nueva variante, C++.

“La velocidad de ejecución de C++ es superior a muchos lenguajes. No es orientado a objetos puro si se compara con otros lenguajes, por ejemplo Java; aunque introduce nuevas palabras claves y operadores para el manejo de clases. Algunas de sus extensiones tienen aplicación fuera del contexto de programación orientada a objetos pues muchos aspectos pueden ser usados independientemente de las clases.” (24)

Es ampliamente utilizado en la construcción de sistemas operativos e intérpretes además de haber sido un punto de partida para la creación de otros lenguajes de alto nivel, entre ellos C# y Java. Constituye un lenguaje que se adapta a múltiples situaciones. Puede ser aplicado en escenarios donde se requiera un bajo nivel en su programación así como situaciones que necesiten de un mayor nivel de abstracción.

No es fuertemente tipado, lo que permite que se realice cualquier conversión entre tipos.

1.8.3 Java

Java constituye un lenguaje maduro, eficaz y muy versátil. Permite que los desarrolladores creen un sistema informático en una plataforma y ejecutarlo en otra distinta de la que se creó inicialmente. Se caracteriza por su potencia, pero a la vez elimina las características menos usadas y más complejas de otros lenguajes como C y C++.

Fue creado por Sun Microsystems el cual controla todo el desarrollo del lenguaje. La mayor parte del código Java se encuentra bajo licencia GPL (Licencia Pública General (traducción al idioma Español) que protege la distribución de software sin restricciones así como su uso y modificación) de GNU (Proyecto

informático creado para la construcción de sistemas operativos libres) excepto las bibliotecas de clases de Sun.

La tecnología Java está compuesta por dos partes fundamentales, el lenguaje de programación y la plataforma de desarrollo. Técnicamente es interpretado aunque necesita de un proceso previo de compilación. Una vez compilado se crea un fichero que almacena el Bytecode (Pseudocódigo prácticamente al nivel de código máquina). Para su ejecución es necesario contar con una Máquina Virtual de Java (JVM), consiste en un intérprete el cual lee el Bytecode y lo traduce en instrucciones ejecutables.

La gestión de punteros y memoria la realiza el propio lenguaje, liberado el programador de estas funciones.

“Con este lenguaje se crean programas tanto para navegadores Web, servicios Web, potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo costo y para cualquier tipo de dispositivo digital.”(25)

“Los programas hechos con Java se caracterizan por ser orientados a objetos, distribuidos, interpretados, robustos, seguros, de arquitectura neutral, portátiles, de gran rendimiento, multitareas y dinámicos.”(26)

“Debido a que la máquina virtual de Java las aplicaciones escritas en este lenguaje tienen un menor rendimiento con relación a las equivalentes escritas en código máquina nativo. Una respuesta a este problema es el empleo de compiladores JIT (Just In Time traducido al español justo a tiempo) Un compilador JIT interacciona con la máquina virtual de Java para convertir el código de bytes en código nativo.”(27)

1.8.4 ¿Por qué utilizar Java?

Se propone utilizar Java como lenguaje en el desarrollo de esta solución informática para que la misma sea multiplataforma. Existen librerías bien documentadas para la Gestión de Interfaces de Usuario. El lenguaje se encuentra bajo la licencia GNU/GPL. Permite el diseño y construcción de aplicaciones de

múltiples propósitos. Se podrá aprovechar las ventajas que ofrece la programación basada en varios hilos de ejecución de un mismo programa.

1.9 Patrones

1.9.1 Patrones de diseño

Patrones GRASP

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Dentro de los patrones de diseño están los GRASP que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades) El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se quiere diseñar eficazmente el software orientado a objetos.(20)

Patrón Experto:

Encargado de asignar una responsabilidad al experto en la información que sería la clase que cuenta con la información necesaria para cumplir la responsabilidad.(20)

Patrón Creador:

Es el encargado de asignarle a la clase B la responsabilidad de crear una instancia de clase A.

La clase B crea las instancias de A si:

- B agrega los objetos de A
- B contiene los objetos de A
- B registra las instancias de los objetos de A.
- B tiene los datos de inicialización que serán enviados a A cuando este objeto sea creado.(20)

Patrón Controlador:

Encargado de asignar la responsabilidad del manejo de mensajes de los eventos del sistema a una clase que represente alguna de las siguientes opciones:

- El sistema global.
- La empresa u organización global.

- Algo activo en el mundo real que pueda participar en la tarea.
- Un manejador artificial de todos los eventos del sistema de un caso de uso (controlador de casos de uso).(20)

Patrón Bajo Acoplamiento:

Encargado de asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. Una clase con bajo acoplamiento no depende de “muchas otras” clases. Las clases con alto acoplamiento recurren a muchas clases y no es conveniente. Son más difíciles de mantener, entender y reutilizar.(20)

Patrón Alta Cohesión:

Encargado de asignar a las clases responsabilidades de forma tal que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad.(20)

Patrones GoF

De creación:

Singleton:

Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. (21)

Estructurales:

Facade:

Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar. (21)

Patrones de Casos de Uso

Patrón CRUD (Creating, Reading, Updating and Deleting):

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

Completo:

Este patrón consta de un caso de uso, llamado Información CRUD o Gestionar información, modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales

como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples.

Parcial:

Este patrón alternativo modela una de las vías de los casos de uso como un caso de uso separado. Es preferiblemente utilizado cuando una de las alternativas de los casos de uso es más significativa, larga o más compleja que las otras.

1.10 Herramientas

En todo proceso de desarrollo de un software se utilizan una serie de herramientas, dentro de ellas están las de modelación, a continuación una breve explicación de algunas de ellas, incluyendo la utilizada en el proyecto ABAD.

1.10.1 Rational Rose

IBM Rational Rose Enterprise es uno de los productos más completos de la familia Rational Rose. Todos los productos de Rational Rose dan soporte a UML, pero no son compatibles con las mismas tecnologías de implementación. Rational Rose Enterprise es un entorno de modelado que permite generar código a partir de modelos Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Al igual que todos los productos de Rational Rose, ofrece un lenguaje de modelado común que agiliza la creación del software.

Incluye también estas funciones:

En general Rational Rose posee las capacidades de modelado con UML, soporta ocho de los nueve diagramas del UML. Rose soporta los lenguajes con excepción de C# y VB.NET, de hecho, Rational ha desarrollado otra herramienta; Rational XDE para .NET, para apuntar al ambiente .NET. En el soporte para el ciclo de vida del proyecto se puede comprar otras herramientas para complementar a Rose, el costo involucrado no siempre hará de ésta una solución aceptable para todas las empresas con excepción de las más grandes. No obstante, no hacen falta otras características específicas que no vengan con las herramientas y el soporte de terceras partes para las herramientas que puede ser un factor importante aquí Rational Rose posee actualmente un soporte muy amplio.

Permite especificar, analizar, diseñar el sistema antes de codificarlo. Es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML1.1.

Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

Rational Rose utiliza un proceso de desarrollo iterativo controlado, donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración. Primero se identifican los riesgos y después se prueba la aplicación para que éstos se hagan mínimos. Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración. Rose permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo. También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades. Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML. Rational Rose proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño. (18)

1.10.2 Visual Paradigm para UML

Visual Paradigm para UML es un producto que facilita a las organizaciones el diseño visual y el diagrama, integrar y desplegar sus aplicaciones empresariales de misión crítica y sus bases de datos subyacentes. La herramienta de desarrollo de software ayuda a su equipo a sobresaltar todo el modelo de construcción; desplegar software de proceso de desarrollo y aumentar al máximo la aceleración de ambos equipos y de los individuos.

Modelado Visual.

- Modelado Visual es el proceso que permite representar gráficamente el sistema software, permitiendo resaltar los detalles más importantes.
- Un buen modelo:

- Identifica requisitos y comunica información.
- Se centra en cómo los componentes del sistema interactúan entre ellos, sin entrar en detalles excesivos.
- Permite ver las relaciones entre los componentes del diseño.
- Mejora la comunicación entre los miembros del equipo usando un lenguaje gráfico.
- Genera modelos que pueden ser utilizados para generar códigos en más de 10 idiomas.

UML2.1habilitado:

El UML es un estándar ampliamente utilizado en la industria de software para el modelado de software. Ayuda a los profesionales a visualizar, comunicar y aplicar sus diseños. Se puede apreciar la evolución de UML a través de los años y actualmente se encuentra en su serie de versiones 2.x. Visual Paradigm para UML ha sido actualizada con rapidez en la sincronización con el nuevo desarrollo de 2,1 UML para proporcionar un entorno de modelado visual que se reúne hoy el software de la tecnología y necesidades de comunicación.

Visual Paradigm ofrece:

- Entorno de creación de diagramas para UML 2.1.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.(19)

Herramienta Case utilizada en el proyecto

1.10.3 ¿Por qué Visual Paradigm?

Visual Paradigm es una herramienta CASE que utiliza “UML” como lenguaje de modelado, es una herramienta de diseño que tiene compatibilidad con el sistema operativo Linux, por lo que es la que se

seleccionó en el proyecto respetando las políticas de trabajo del Alba, que están centradas en trabajar sobre herramientas libres. Tiene la ventaja de ser multiplataforma, soporta el ciclo de vida completo del desarrollo de software. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas, documentar con mayor exactitud todo el trabajo efectuado, realizar un diseño centrado en casos de uso y enfocado al negocio, modelar el sistema evidenciándose en los modelos de análisis y diseño que se presenta, todo esto permitiendo obtener exitosos resultados.

A diferencia de las herramientas de modelado de empresa en el mercado, Visual Paradigm UML emplea para una respuesta rápida y bajos requisitos de memoria del motor de persistencia, lo que le permite manejar grandes y complicadas estructuras de proyecto en una manera muy eficiente y sin embargo, sólo requiere una configuración de escritorio. Lo que no es posible con Rational Rose por la gran necesidad de memoria de esta herramienta.

Conclusiones

En el desarrollo de este capítulo se realizó un estado del arte de las principales técnicas usadas por los arquitectos de información, además se explicó las características e importancia de las Herramientas CASE para mejorar los procesos informáticos.

Como resultado se llegó a las siguientes conclusiones:

Las herramientas existentes no recogen las técnicas Análisis de Secuencia y Control de Términos. Por este motivo es necesario realizar los flujos de negocio, requisitos, análisis y diseño a estas dos técnicas para su posterior implementación.

Se explicó el motivo de la selección de RUP como metodología de desarrollo de software atendiendo a facilidades como: su factibilidad en proyectos de larga duración, así como que el reconocimiento de las necesidades del usuario y sus requisitos no se pueden definir completamente en una primera iteración. Además de la facilidad que brinda de tener la documentación archivada para las posteriores versiones de la herramienta.

Se utiliza BPM para el modelado del negocio, ya que a los clientes le resulta más fácil comprender el desarrollo de un proceso mediante un diagrama de flujos como se propone en BPM, que la terminología de casos de uso de negocio planteada por RUP.

Como herramienta de modelado se escogió Visual Paradigm por su disponibilidad en múltiples plataformas, tener compatibilidad con Linux y emplear una respuesta rápida y bajos requisitos de memoria del motor de persistencia.

Como lenguaje de programación a utilizar en el proceso de desarrollo de software se escogió Java para que la aplicación sea multiplataforma, por tener librerías bien documentadas para la gestión de Interfaz de Usuario, además de que el lenguaje se encuentra bajo la licencia de GNU/GPL.

CAPÍTULO 2: Características del Sistema

Introducción

En este capítulo se realiza la descripción de los procesos del negocio utilizando BPM, además de la Ingeniería de Requisitos utilizando tres etapas: la elicitación o captura de requisitos para orientar el desarrollo del software por el camino al cual se quiere llegar con el sistema a construir, el análisis de requisitos para determinar el dominio de la aplicación, el desempeño del sistema, así como las restricciones que el mismo debe poseer, la especificación de los requisitos para determinar lo que el sistema debe realizar, además de las descripciones de los casos de uso para mayor comprensión de su funcionamiento y el esclarecimiento del diseño del sistema que se desea construir. Luego de realizar todas estas actividades se obtienen los artefactos generados a partir de los flujos de trabajo de negocio y requisitos.

2.1 Características del Sistema

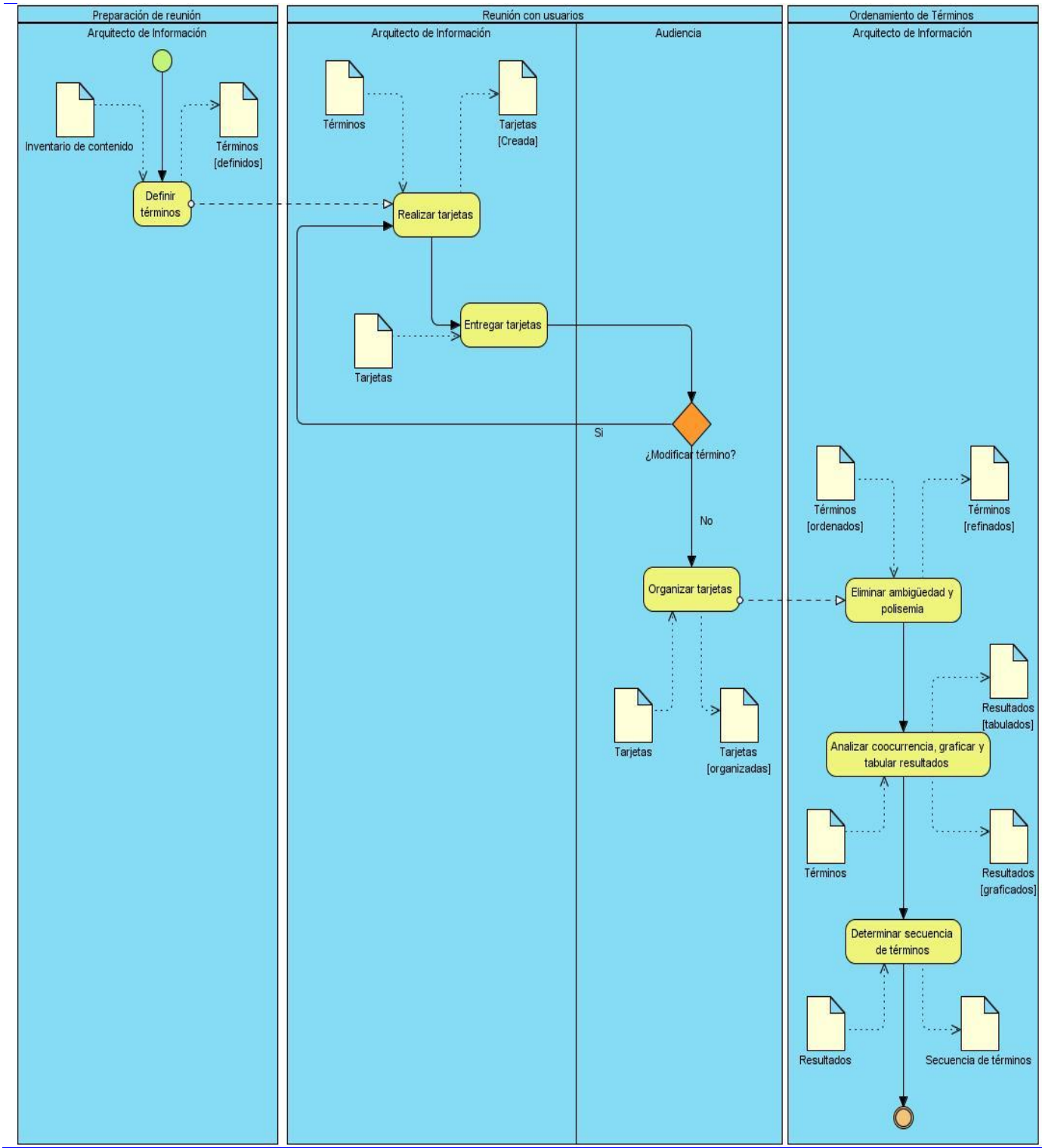
Estas técnicas consisten en la aplicación del análisis de coocurrencia para cuantificar resultados y hacer precisa la toma de decisiones. Con la aplicación de estas técnicas se logran crear secuencias de términos y definir los términos que se corresponden con el modelo mental de los usuarios.

2.2 Modelos de procesos del Negocio utilizando BPM

2.2.1 Modelo de Procesos del Negocio de la Técnica Análisis de Secuencia

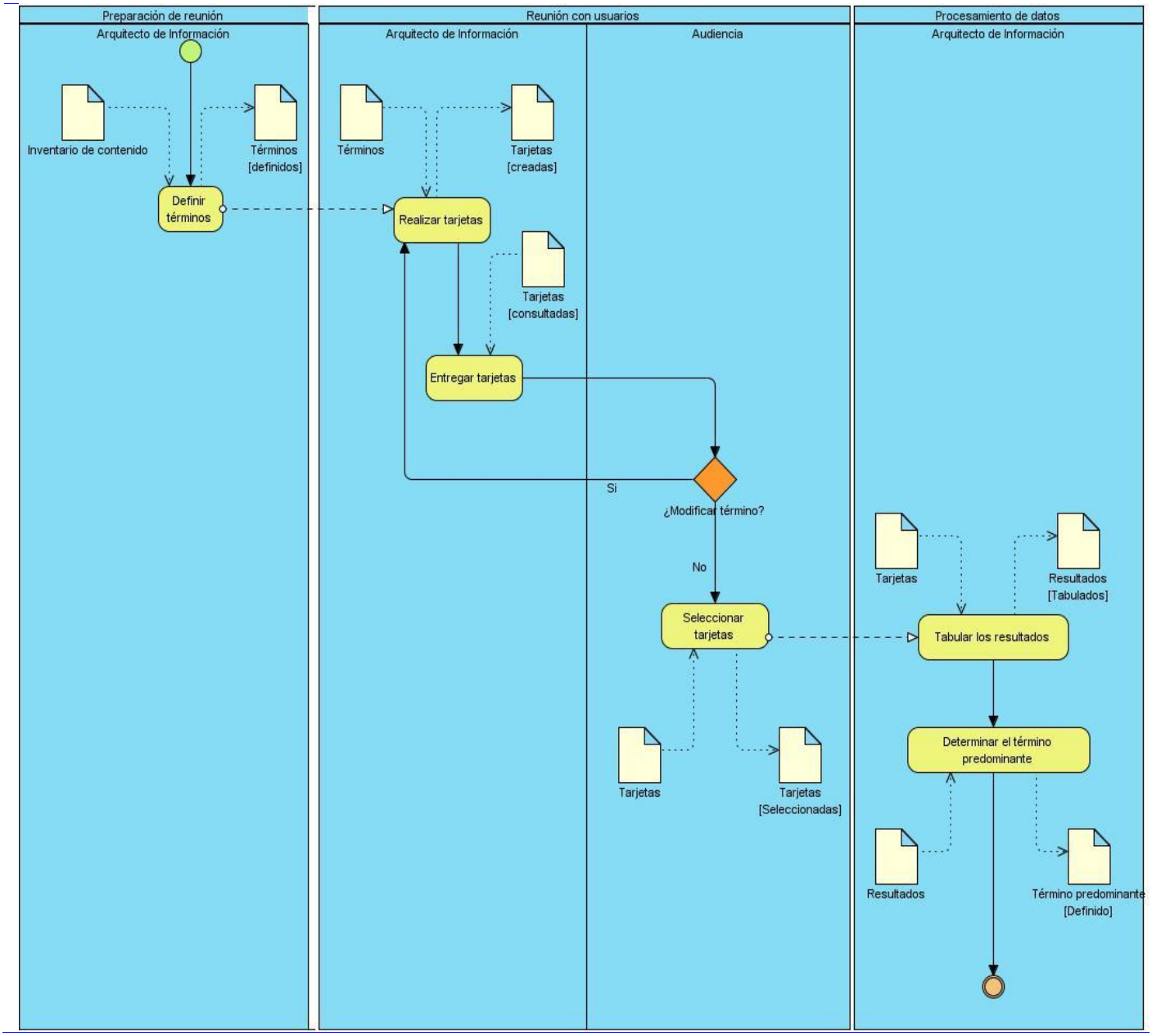
Ficha de Proceso	
Proceso:	Realizar la técnica de Análisis de Secuencia.
Descripción:	Es el proceso mediante el cual se realizan una serie de pruebas a usuarios potenciales del producto, y el posterior análisis cualitativo y cuantitativo de esos resultados para ayudar a definir la secuencia de las etiquetas en el producto electrónico.
Entradas:	Posibles términos.
Salidas:	Términos organizados.
Reglas del Negocio:	Se necesitan representar en un producto electrónico términos para el etiquetado de forma ordenada.

Actividades Principales	
Actividad 1	Definir términos: El Arquitecto de Información a través del inventario de contenido define los posibles términos.
Actividad 2	Realizar tarjetas: El Arquitecto de Información prepara las tarjetas que serán entregadas a la audiencia.
Actividad 3	Entregar tarjetas: El Arquitecto de Información entrega las tarjetas a la audiencia que fueron preparadas con antelación para que éstos las organicen.
Actividad 4	Organizar tarjetas: La audiencia selecciona la secuencia de términos según su criterio.
Actividad 5	Eliminar ambigüedad y polisemia: El Arquitecto de Información analiza los resultados eliminando la ambigüedad y la polisemia.
Actividad 6	Analizar coocurrencia, graficar y tabular resultados: El Arquitecto de Información después de analizar los resultados los grafica y los tabula.
Actividad 7	Determinar secuencia de términos: El Arquitecto de Información obtiene la secuencia de términos luego de analizar los resultados.
Diagrama de Proceso	


Preparación de reunión
Arquitecto de Información
Inventario de contenido
Términos [definidos]
Definir términos
Reunión con usuarios
Arquitecto de Información
Audiencia
Términos
Tarjetas [Creada]
Realizar tarjetas
Entregar tarjetas
Tarjetas
Si
¿Modificar término?
No
Organizar tarjetas
Tarjetas
Tarjetas [organizadas]
Ordenamiento de Términos
Arquitecto de Información
Términos [ordenados]
Términos [refinados]
Eliminar ambigüedad y polisemia
Resultados [tabulados]
Analizar coocurrencia, graficar y tabular resultados
Términos
Resultados [graficados]
Determinar secuencia de términos
Resultados
Secuencia de términos

2.2.2 Modelo de Procesos del Negocio de la Técnica Control de Términos

Ficha de Proceso	
Proceso:	Realizar la técnica de Control de Términos.
Descripción:	Es el proceso mediante el cual se realizan de una serie de pruebas a usuarios potenciales del producto, y el posterior análisis para ayudar a definir las etiquetas del producto electrónico.
Entradas:	Posibles términos.
Salidas:	Término predominante.
Reglas del Negocio:	Se necesita escoger un término para el etiquetado de un producto electrónico.
Actividades Principales	
Actividad 1	Definir términos: El Arquitecto de Información a través del inventario de contenido define los posibles términos.
Actividad 2	Realizar tarjetas: El Arquitecto de Información prepara las tarjetas que serán entregadas a la audiencia.
Actividad 3	Entregar tarjetas: El Arquitecto de Información entrega las tarjetas a la audiencia que fueron preparadas con antelación para que éstos las organicen.
Actividad 4	Seleccionar tarjetas: La audiencia selecciona el término dominante según su criterio.
Actividad 5	Tabularlos resultados: El Arquitecto de Información después de analizar los resultados los tabula.
Actividad 6	Determinar el término predominante: El Arquitecto de Información obtiene el término predominante luego de analizar los resultados.
Diagrama de Proceso	



2.3 Requisitos Funcionales

Teniendo en cuenta que la aplicación se encarga de gestionar información sobre tarjetas, usuarios y los ejercicios de cada una de las técnicas, se incluyó el concepto Requisito de Información (RI) que describe qué información debe almacenar el sistema para satisfacer las necesidades de los usuarios, además identifica los conceptos relevantes sobre los datos específicos que son de interés.(31)

Los Objetivos de la aplicación son:

- Gestionar las Tarjetas.
- Gestionar el Ejercicio de la Técnica Análisis de Secuencia.
- Gestionar el Ejercicio de la Técnica Control de Términos.
- Gestionar Participante.
- Visualizar Resultados.

Los Requisitos de Información definidos para el sistema se describen a continuación:

RI-1	Información sobre Tarjetas.
Objetivos asociados	OBJ-1Gestionar Tarjetas
Requisitos Asociados	RF 01. Mostrar listado de las tarjetas. RF 02. Adicionar tarjetas. RF 03. Modificar tarjetas.
Descripción	El sistema debe permitir la gestión de las tarjetas como parte del proceso de Crear un Ejercicio de Técnicas.
Datos específicos	• Datos del listado: Términos (nombre) y número. • Datos de las tarjetas: Términos (nombre) y número.

RI-2	Información sobre un Ejercicio de la Técnica Análisis de Secuencia.
Objetivos asociados	OBJ-2 Gestionar el Ejercicio de la Técnica Análisis de

	Secuencia
Requisitos Asociados	RF 4.Crear un Ejercicio dela Técnica Análisis de Secuencia. RF 5. Modificar un Ejercicio de la Técnica Análisis de Secuencia. RF 6. Ejecutar un Ejercicio de la Técnica Análisis de Secuencia. RF 7. Guardar Ejercicio de la Técnica Análisis de Secuencia desarrollado. RF 8. Cargar los ficheros de los Ejercicios de la Técnica Análisis de Secuencia resueltos.
Descripción	El sistema debe permitir la gestión de un Ejercicio de la Técnica de Análisis de Secuencia.

RI-3	Información sobre un Ejercicio de la Técnica Control de Términos.
Objetivos asociados	OBJ-3 Gestionar el Ejercicio de la Técnica Control de Términos.
Requisitos Asociados	RF 9. Crear un Ejercicio de la Técnica Control de Términos. RF 10. Modificar un Ejercicio de la Técnica Control de Términos. RF 11. Ejecutar un Ejercicio de la Técnica Control de

	Términos. RF 12. Guardar Ejercicio de la Técnica Control de Términos desarrollado. RF 13. Cargar los ficheros de los Ejercicios de la Técnica Control de Términos resueltos.
Descripción	El sistema debe permitir la gestión de un Ejercicio de la Técnica de Control de Términos.

RI-4	Información sobre Participantes.
Objetivos asociados	OBJ-4 Gestionar Participante
Requisitos Asociados	RF 14. Adicionar Participante. RF 15. Modificar Participante. RF 16. Eliminar Participante. RF 17. Listar Participantes.
Descripción	El sistema debe permitir la gestión de los participantes.
Datos específicos	• Datos del Participante: Nombre y correo electrónico.

Para reflejar los requisitos funcionales que no entran dentro de la clasificación de RI se utilizó una tabla que contiene el requisito con su nomenclador de la forma RF- #, la descripción del requisito con todos los datos implicados y el caso de uso con el que se le da seguimiento. Los que se presentan a continuación no están contenidos dentro de los requisitos de información descritos anteriormente.

RF-18	Visualizar resultados.
Descripción	El sistema debe permitir la visualización de gráficas y tablas de cada una de las técnicas aplicadas con los resultados.
Seguimiento	CUS Visualizar_Resultados

2.4 Requisitos no Funcionales.

Hardware

- Microprocesador: 2 GHz como mínimo.
- Memoria RAM: 512 MB.
- PC PENTIUM IV o superior.
- Impresora.

Software

- Sistema Operativo: Windows 2000 o superior y Linux.

Seguridad

- Cada usuario accederá sólo a la información que le corresponda según su nivel de acceso.
- El sistema debe recuperase ante fallos, por perdida de alimentación.
- Se llevará a cabo tratamiento de excepciones.

Portabilidad

- La aplicación deberá correr sobre cualquier Sistema Operativo.

Restricciones de diseño e implementación

- Uso de plataforma libre.
- Se utilizará el lenguaje de programación Java.
- Se desarrollará el sistema sobre el IDE NetBeans Versión 6.7 que puede correr en cualquier plataforma.
- Empleo de la filosofía de la Programación Orientada a Objetos (POO).

2.5 Actores del Sistema

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

Actor del Sistema	Descripción
Arquitecto de Información	Persona encargada de analizar y elaborar el informe de recomendaciones para la organización de la información. Además, es el encargado de preparar y ejecutar el ejercicio a desarrollar por la audiencia.

2.6 Estructuración del Diagrama de Casos de Uso del Sistema.

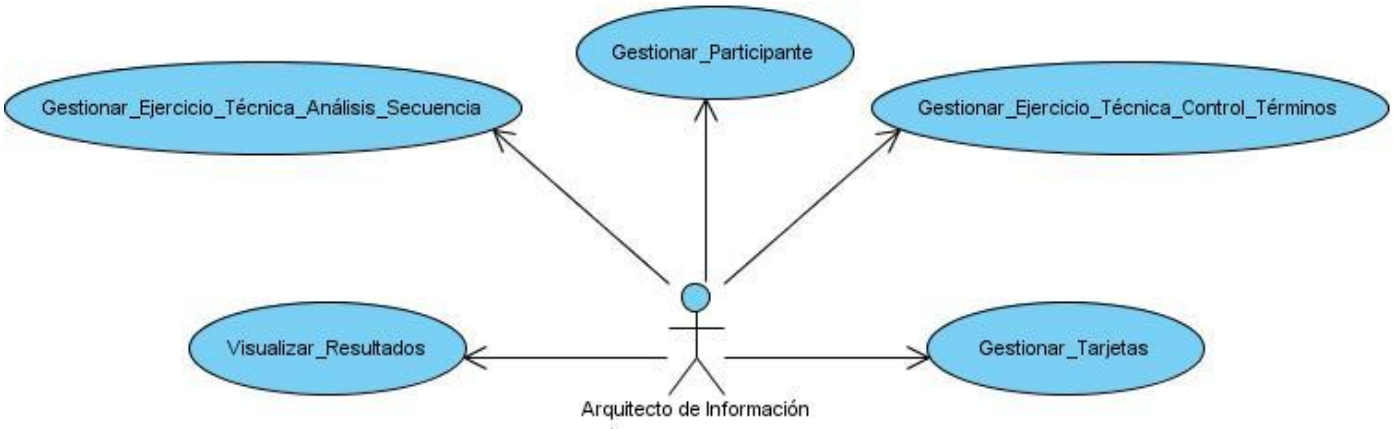


Figura 2: Diagrama de Casos de Uso del Sistema

2.6.1 Especificación textual de los casos de uso.

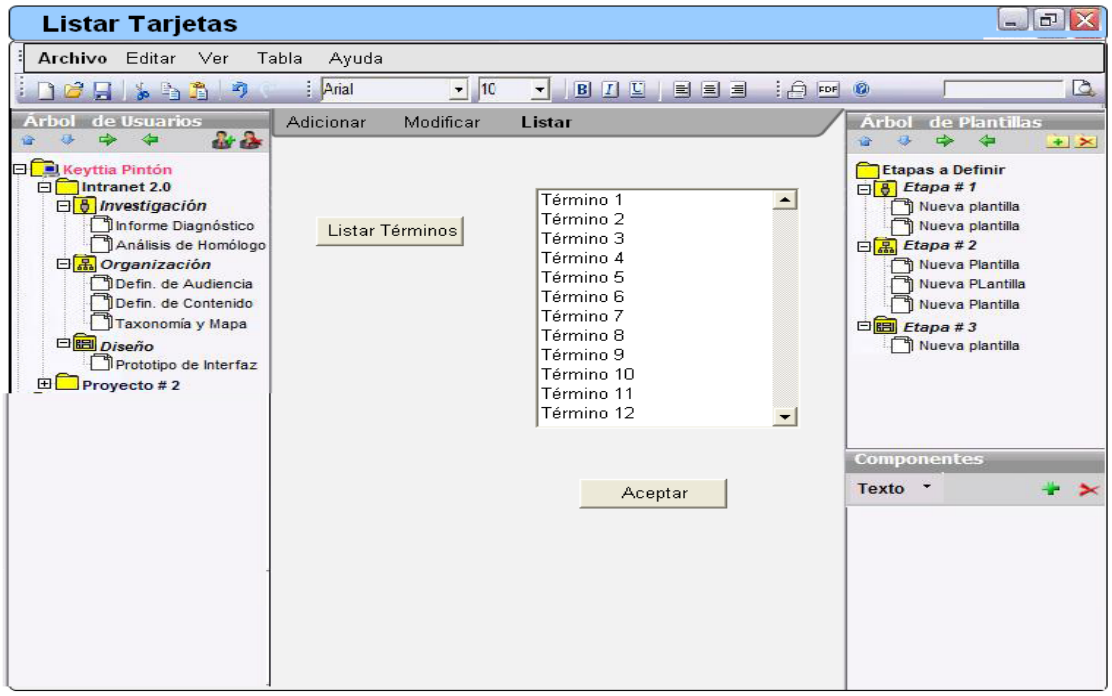
Las especificaciones de los casos de uso contienen las propiedades textuales de cada uno de ellos. Se utilizan junto a una herramienta de gestión de requisitos para especificar y marcar los requisitos dentro de las propiedades de los casos de uso. Cada caso de uso hace referencia al prototipo de interfaz correspondiente.

A continuación la descripción textual perteneciente al CU: Gestionar_Tarjeta.

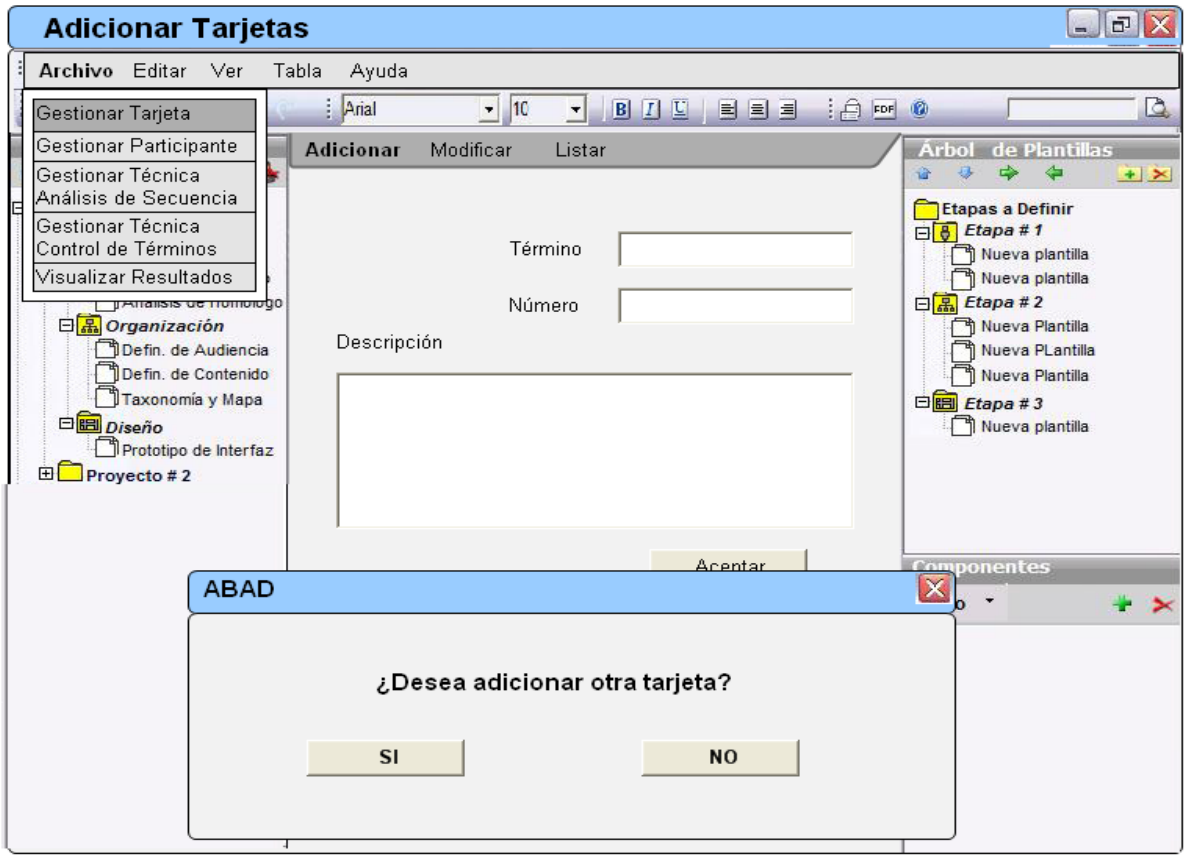
Caso de Uso:	Gestionar_Tarjetas
Actores:	Arquitecto de Información
Resumen:	El CUS se inicia cuando el Arquitecto de Información selecciona la opción de gestionar tarjetas, luego selecciona el tipo de gestión, el sistema realiza la acción seleccionada por el Arquitecto de Información y finaliza el caso de uso.
Precondiciones:	Que exista la tarjeta.
Referencias	RF 01, RF 02, RF 03
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Arquitecto de Información elige la opción de gestionar tarjetas.	1.1. El sistema muestra las opciones de mostrar listado de tarjetas, adicionar tarjetas y modificar tarjetas.

Sección "Listar Tarjetas"	
Acción del Actor	**Respuesta del Sistema**
1. El Arquitecto de Información selecciona la opción de listar tarjetas.	1.1. El sistema muestra el listado de las tarjetas y finaliza el caso de uso.
Prototipo de Interfaz	

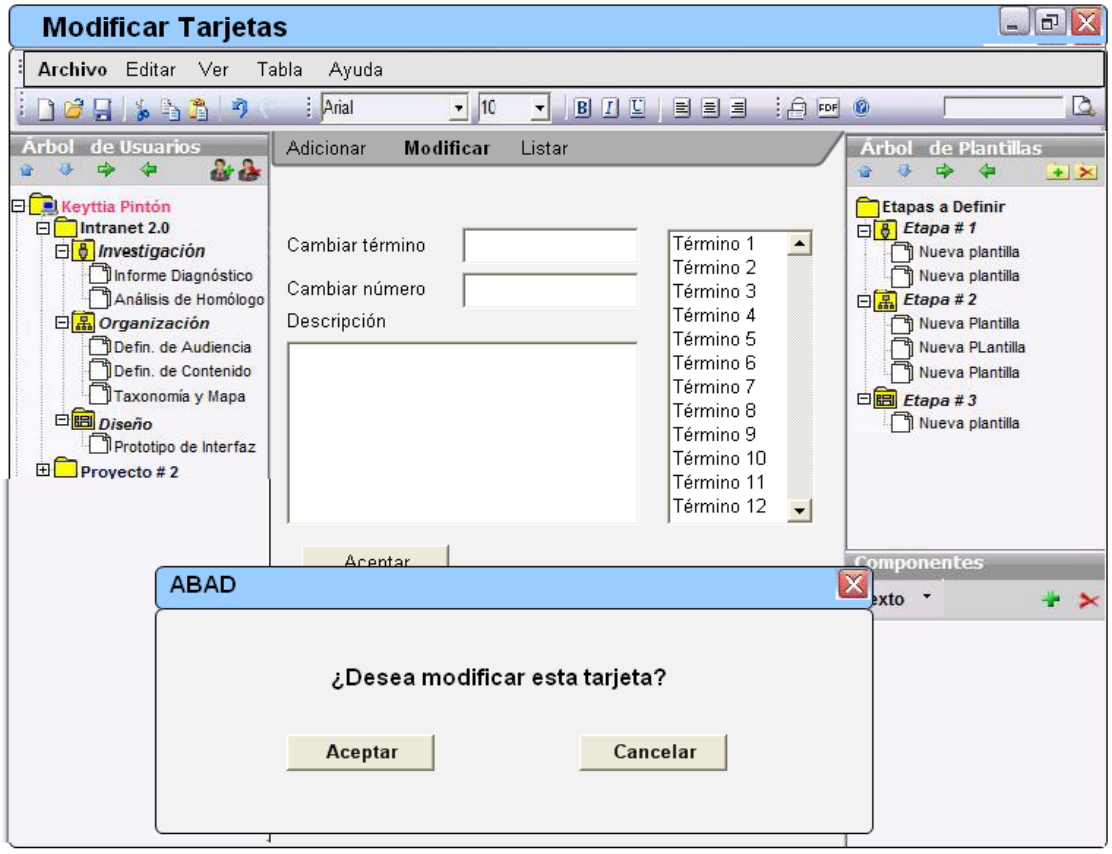


Sección "Adicionar Tarjetas"	
Acción del Actor	**Respuesta del Sistema**
1. El Arquitecto de Información selecciona la opción de adicionar tarjetas.	1.1. El sistema permite adicionar la nueva tarjeta.
2. El Arquitecto de Información inserta el nombre de la tarjeta y selecciona la opción aceptar.	2.1. Si el Arquitecto de Información selecciona la opción aceptar se adiciona la tarjeta y el sistema muestra un mensaje preguntando si se desea adicionar otra tarjeta.
3. El Arquitecto de Información selecciona la opción sí o no.	3.1. El sistema vuelve a mostrar la opción de gestionar tarjetas.
Prototipo de Interfaz	



Flujos Alternos	
Flujo alterno al paso 3 "Cancelar operación"	
Acción del Actor	**Respuesta del Sistema**
	3.1. El sistema vuelve a la interfaz de gestionar tarjeta y finaliza el caso de uso.
Sección "Modificar Tarjetas"	

Acción del Actor	Respuesta del Sistema
1. El Arquitecto de Información selecciona la opción de modificar tarjetas.	1.1. El sistema muestra un listado de las tarjetas del ejercicio.
2. El Arquitecto de Información selecciona la tarjeta a modificar.	2.1. El sistema muestra un formulario para modificar la tarjeta.
3. El Arquitecto de Información cambia los datos de la tarjeta y selecciona la opción aceptar.	3.1. El sistema muestra un mensaje para confirmar la acción.
4. El Arquitecto de Información selecciona la opción aceptar o cancelar.	4.1. El sistema modifica la tarjeta con el dato nuevo y finaliza el caso de uso.
Prototipo de Interfaz	



Flujos Alternos	
Flujo alterno al paso 4 "Cancelar operación"	
	4.1. El sistema cancela la acción y finaliza así el caso de uso.
Pos-condiciones	Listado de tarjetas mostrado. Tarjetas adicionadas. Tarjetas modificadas.

Consultar las descripciones de los Casos de Uso en el artefacto Modelo de Sistema.

Conclusiones

Después de realizada la descripción de los procesos de negocio utilizando BPM, la estructuración del diagrama de CUS y la especificación de los casos de uso, como resultado de las actividades de la Ingeniería de Requisitos se concluye que:

- El elemento principal que permitió la captura de todas las necesidades de la parte interesada fue la elicitación utilizando técnicas de levantamiento de requisitos como: entrevistas, arqueología de documentos y tormenta de ideas. Se pudieron estructurar por objetivos del sistema los requisitos a través del análisis. Se facilitó el entendimiento de las funcionalidades y restricciones del sistema durante la especificación de los requisitos obtenidos.
- Se logró realizar de forma clara la descripción de los casos de uso, lo que permitirá hacer el análisis y diseño de dichas técnicas.

CAPÍTULO 3: Análisis y Diseño de las técnicas Análisis de Secuencia y Control de Términos.

Introducción

En el presente capítulo se continuará con el flujo de trabajo Análisis y Diseño en el que se obtendrán los principales artefactos para la implementación de las técnicas Análisis de Secuencia y Control de Términos. En el análisis definido por la metodología de desarrollo escogida se tendrán en cuenta los requisitos funcionales para la confección de los diagramas de clases y los diagramas de colaboración correspondientes a cada caso de uso. En el diseño consolidará el análisis propuesto con los diagramas de clases, los diagramas de secuencia correspondientes a cada escenario y la aplicación de patrones de arquitectura y de diseño como una manera más práctica de describir ciertos aspectos.

3.1 Análisis

En la etapa de Análisis se realiza un modelo de análisis donde se incluyen los diagramas de clases. Esto proporciona la estructura de una vista interna del sistema, estructurado por clases estereotipadas: Entidad: Modelan información que posee larga vida y que es a menudo persistente. Interfaz: Modelan la interacción entre el sistema y sus actores. Controladora: Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.

El análisis es utilizado fundamentalmente por los desarrolladores para comprender cómo debe estructurarse el sistema, es decir, de qué forma debe ser diseñado e implementado. No debe contener redundancias o inconsistencias entre los requisitos. Sirve como una primera aproximación del diseño.

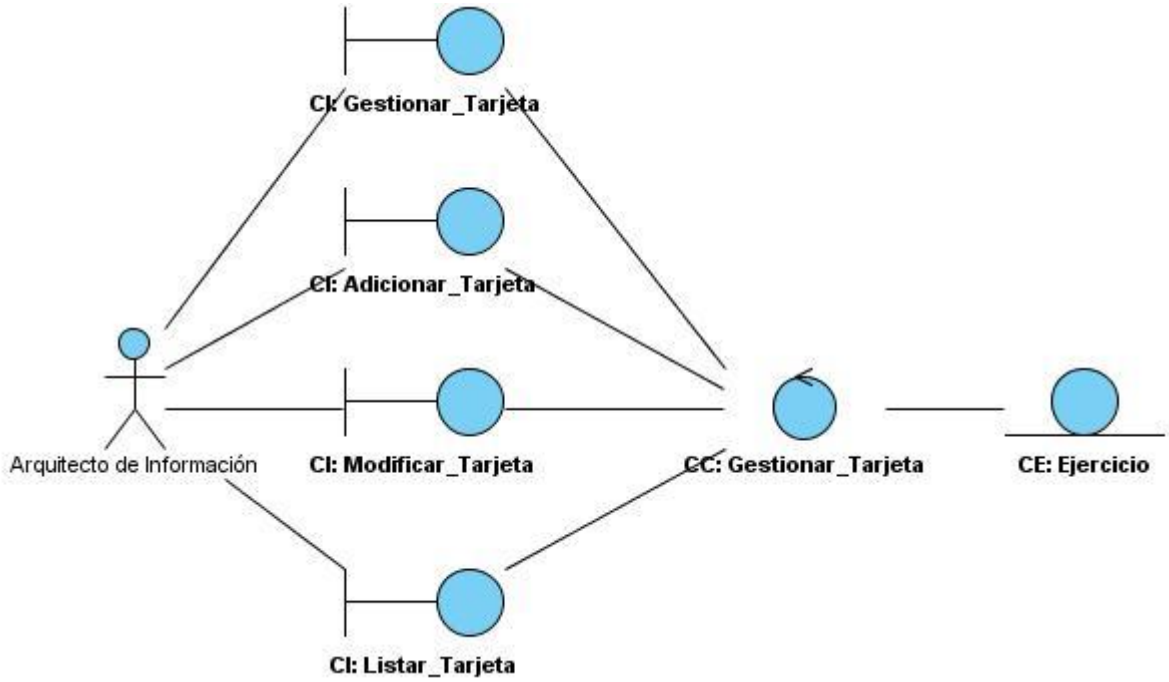
En el análisis podemos estructurar los requisitos de manera que nos facilite su comprensión, su preparación, su modificación y en general su mantenimiento. Esta estructura basada en clases de análisis es independiente de la estructura que se dio a los requisitos. Sin embargo, existe una trazabilidad directa entre esas distintas estructuras, la cual se define entre casos de uso del modelo de casos de uso y realizaciones de casos de uso en el modelo de análisis.

3.1.1 Diagramas de Clases del Análisis

Las clases del análisis se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen

relaciones (asociación, agregación-composición, generalización-especialización y tipos asociativos). Los tipos de clases propuestas por RUP son: interfaz, control y entidad.

A continuación el diagrama de clases del análisis perteneciente al CU: Gestionar_Tarjeta.



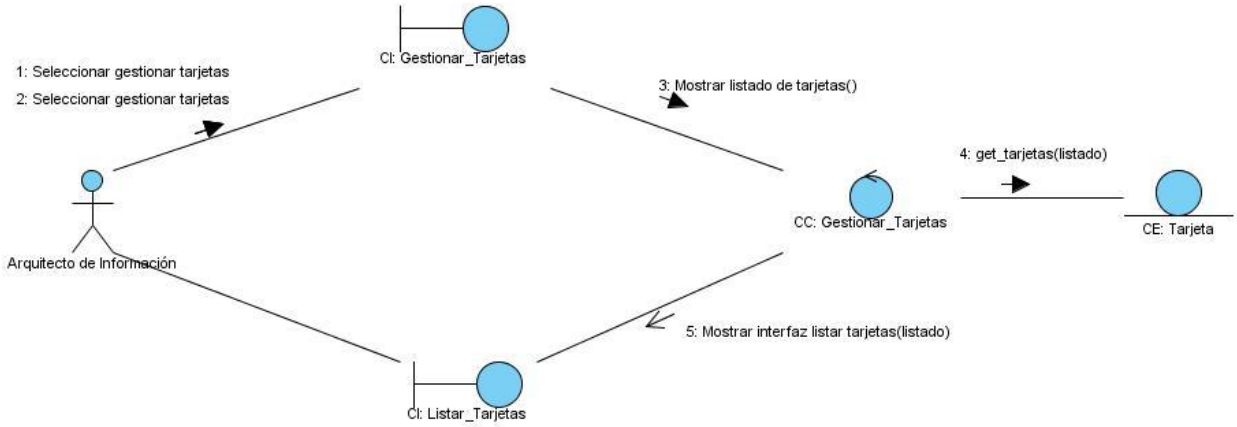
Consultar los diagramas de Clase del Análisis restantes en el artefacto Modelo de Análisis.

3.1.2 Realización de casos de uso del análisis

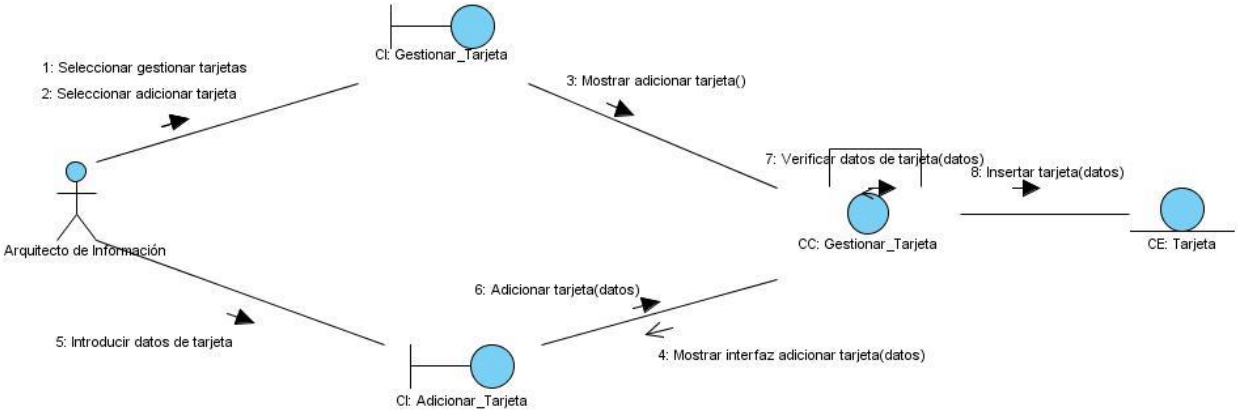
Para la realización de los casos de uso del análisis propuesto para el módulo Técnicas se utilizarán los diagramas de interacción. De dichos diagramas se escogieron los de colaboración, los cuales destacan la organización estructural de los objetos que envían y reciben mensajes, ofrecen una visión clara del flujo de control en el contexto en el que se desarrollan, son útiles en la fase exploratoria para identificar objetos y la estructura estática viene dada por los enlaces; la dinámica por el envío de mensajes.

A continuación los diagramas de colaboración pertenecientes al CU: Gestionar_Tarjeta.

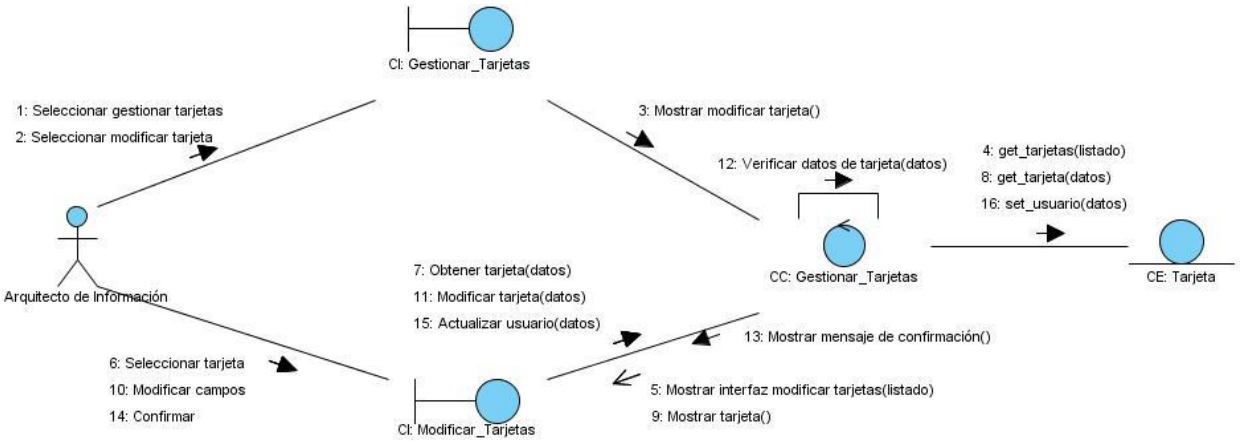
Sección: Listar Tarjetas.



Sección: Adicionar Tarjetas.



Sección: Modificar Tarjetas.



Consultar los diagramas de Colaboración en el artefacto Modelo de Análisis.

3.2 Diseño

En la etapa de Diseño se realiza el modelo de diseño que está muy cercano al de implementación. Se puede guardar y mantener el modelo de diseño a través del ciclo de vida del software. El propósito fundamental del diseño es crear una entrada apropiada y un punto de partida para actividades de implementación. Es un modelo de objetos que describe la realización física de los casos de uso haciendo uso de los requisitos funcionales y centrándose en los no funcionales, que plantean restricciones relacionadas con el entorno de implementación, lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia, y tecnologías de interfaz de usuario que tienen impacto en el sistema a considerar. Durante esta etapa se generan todas las especificaciones para la programación del sistema.

3.2.1 Arquitectura definida para el sistema

Un estilo arquitectónico o variante arquitectónica define a una familia de sistemas informáticos en términos de su organización estructural. Un estilo arquitectónico describe componentes y las relaciones entre ellos con las restricciones de su aplicación, la composición asociada y el diseño para su construcción.

Arquitectura en capas

Este patrón define cómo organizar el modelo de diseño en capas, que pueden estar físicamente distribuidas, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la

comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. Además, ayuda a identificar qué puede reutilizarse, y proporciona una estructura que ayuda a tomar decisiones sobre qué partes comprar y qué partes construir.(32)

Principales estilos de arquitecturas estratificadas de las aplicaciones distribuidas contemporáneas:

- Arquitecturas de dos niveles
- Arquitecturas de tres niveles
- Arquitecturas de n niveles

Se utilizará la arquitectura en dos niveles:

El primer nivel:

Se denomina capa de presentación y normalmente consiste en una interfaz gráfica de usuario de algún tipo. El segundo nivel, o nivel de empresa, consiste en la aplicación o lógica de empresa.(32)

El segundo nivel (lógica de aplicación):

Es básicamente el código al que recurre el nivel de presentación para recuperar los datos deseados. El nivel de presentación recibe entonces los datos y los formatea para su presentación. Esta separación entre la lógica de aplicación de la interfaz de usuario añade una enorme flexibilidad al diseño de la aplicación. Pueden construirse y desplegarse múltiples interfaces de usuario sin cambiar en absoluto la lógica de aplicación siempre que está presente una interfaz claramente definida en el nivel de presentación.(32)

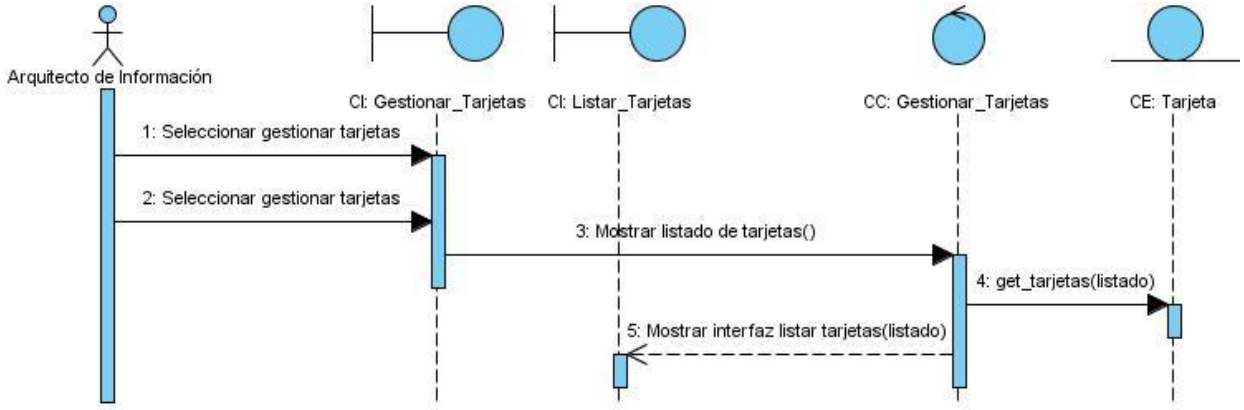
3.2.2 Realización de Casos de Uso

La realización de caso de uso en las clases de diseño es una colaboración que describe cómo se realiza un caso de uso específico, y cómo se ejecuta, en términos de clases de diseño y sus objetos. Proporciona una traza directa a una realización de casos de uso-análisis en el modelo de análisis.

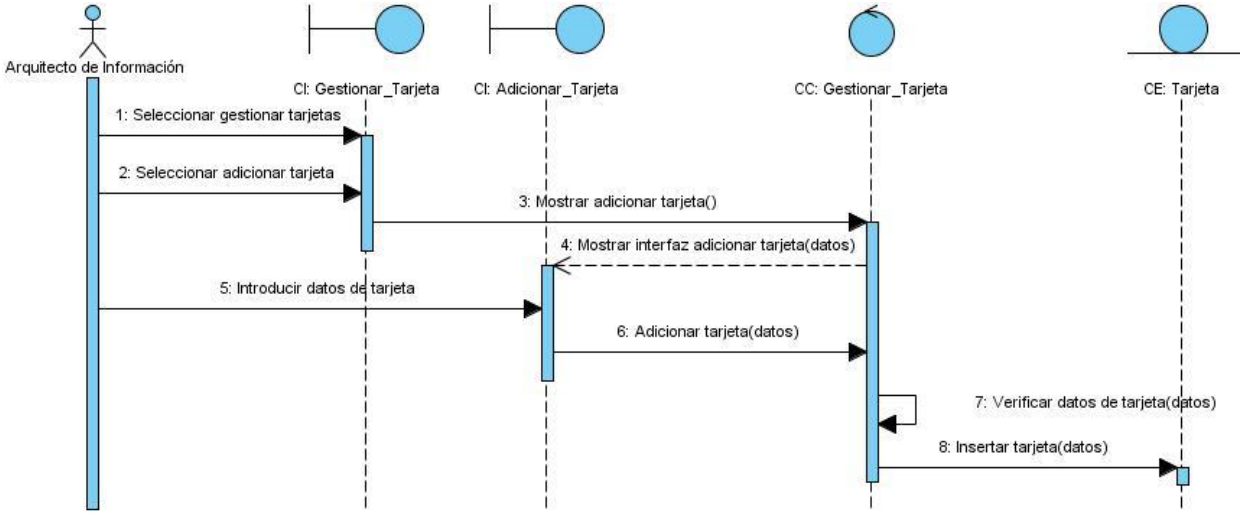
Se seleccionaron los diagramas de secuencia para lograr un mayor nivel de detalle en el diseño en cuanto a la realización de los casos de uso. Estos diagramas permiten reflejar de forma más sencilla el funcionamiento del caso de uso siendo similares a los diagramas de colaboración.

A continuación los diagramas de secuencia pertenecientes al CU: Gestionar_Tarjeta.

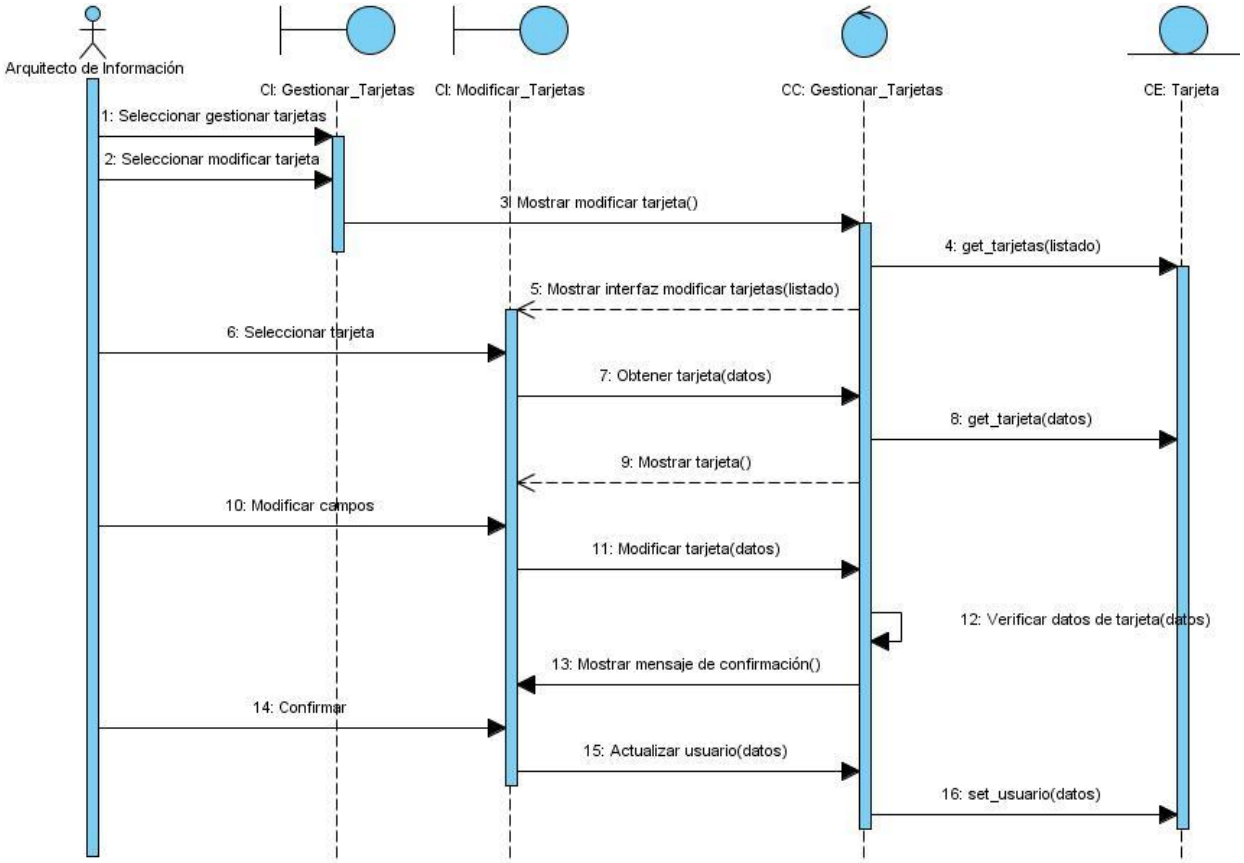
Sección: Listar Tarjetas.



Sección: Adicionar Tarjetas.



Sección: Modificar Tarjetas.



Consultar los diagramas de Secuencia en el artefacto Modelo de Diseño.

Conclusiones

En este capítulo se modeló cada uno de los casos de uso que dan respuesta a las necesidades planteadas por el cliente en la etapa de concepción. Se obtuvo el análisis y diseño del módulo Técnicas con el objetivo de facilitar el trabajo posterior que realizará el equipo de desarrollo. Se propuso el diseño de un prototipo de interfaz de usuario quedando así en el presente capítulo la base para la futura implementación.

CAPÍTULO 4: Validación

Introducción

Las métricas son fundamentales para la evaluación de los productos y procesos de software en los distintos dominios de aplicación, abarcando un amplio contexto en cuanto a medidas de software se refiere. Las métricas son aplicables a un sin número de procesos y productos que estén relacionados y afecten a la estimación de costo.

Como es de esperar las mediciones en el mundo del software son muy importantes y éstas pueden dividirse fundamentalmente en dos grandes categorías: las mediciones directas y las mediciones indirectas. En la categoría de mediciones directas se pueden agrupar: el costo, las líneas de código producidas, velocidad de ejecución, espacio en memoria, así como los defectos observados en un período de tiempo determinado. Por su parte en la categoría de mediciones indirectas podemos agrupar: la funcionalidad, la calidad, la complejidad, la eficiencia, la fiabilidad, la facilidad de mantenimiento, etc.

Se puede afirmar que las métricas del software que están íntimamente relacionadas con el desarrollo de éste son: la complejidad, la funcionalidad y la eficiencia y dentro de ellas podemos encontrar las métricas técnicas, las métricas de calidad y las métricas de productividad, las orientadas a la persona, las orientadas al tamaño y a la función, así como al establecimiento de la estimación como actividad crucial del proceso de gestión.

Concluido el trabajo se obtuvieron artefactos que abarcan desde la fase de inicio hasta las etapas de análisis y diseño de software. Para validar las actividades realizadas y los artefactos obtenidos se emplearon las métricas de la calidad de especificación de requisitos que propone Pressman, las métricas de diseño arquitectónico propuestas por Card y Glass para determinar la complejidad de los módulos y las métricas orientadas a clases.

4.1 Proceso o fase inicial. Modelo de métricas

La estructura y dinámica del entorno organizacional se pudo comprender con la realización de los procesos del negocio. Se pudo identificar y proponer mejoras en los requisitos del sistema que posibilitó minimizar el esfuerzo de los especialistas.

4.1.1 Métrica de la calidad de especificación de los requisitos

Existe toda una lista de características que sugieren el uso de una o más métricas para realizar la validación de los requisitos como son: especificidad (ausencia de ambigüedad), corrección, compleción, comprensión, capacidad de verificación, consistencia externa e interna, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización.

Dadas las particularidades de los requisitos capturados, se aplicará la métrica que mide la especificidad en los mismos, para lograr que los clientes puedan entender los requisitos de una manera fácil y puedan ser probados.

Para llevar a cabo se tiene que cumplir que: $n_r = n_f + n_{nf}$

Donde:

n_r representa el número de requisitos del sistema.

n_f es el número de requisitos funcionales.

n_{nf} es el número de requisitos no funcionales.

Luego la especificidad de los requisitos se mide utilizando la siguiente fórmula:

$Q = n_{ui} / n_r$

Donde:

n_{ui} es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

A medidas que el valor Q se acerca a 1, esto implica que va disminuyendo la ambigüedad de la especificación.

Para obtener la menor ambigüedad posible y mayor claridad en los requisitos, para la evaluación de la métrica propuesta se hicieron 2 revisiones, con el objetivo de cubrir todas las necesidades de los clientes y que éste quedara lo más satisfecho posible.

En la primera revisión realizada se detectó que algunos de los requisitos funcionales y no funcionales, no se correspondían con su redacción presentando problemas de ambigüedad.

De los requisitos funcionales el 0,10 % presentaban ambigüedad, no siendo así con los requisitos no funcionales. Por lo tanto, para un total de 31 requisitos, los revisores técnicos tuvieron la misma interpretación para 28 de ellos.

Q= 28 / 31

Q= 0,90

En una segunda revisión realizando un estudio de los requisitos mucho más profundo, se detectaron menos problemas que en la anterior, encontrándose sólo el 0,03 % de los requisitos funcionales. Por lo que para un total de 31 requisitos, los revisores tuvieron igual interpretación para 30 de ellos. O sea:

Q= 30 / 31

Q= 0,97

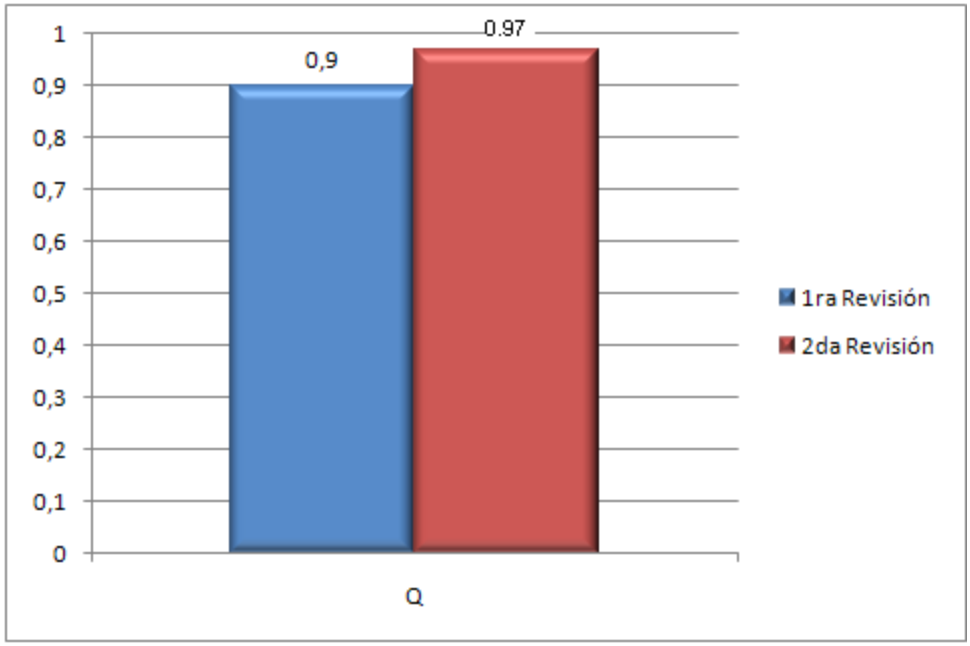


Figura 3: Gráfico de control de la calidad de la especificación de los requisitos

El estudio cuantitativo de los resultados obtenidos en las revisiones, permite llegar a la conclusión de que los requisitos que se obtienen, en su mayoría presentan un grado de ambigüedad lo suficientemente bajo, con un valor de Q muy cercano a 1, siendo suficiente para aceptar este nivel de calidad para los requisitos, artefacto éste más que fundamental para desarrollar con éxito un software que sea capaz de cumplir con las expectativas y necesidades de los clientes.

4.1.2 Aceptación del cliente

Luego de concluir el proceso de análisis y diseño de las técnicas Análisis de Secuencia y Control de Términos pertenecientes al Módulo Técnicas para la plataforma de arquitectura de la información ABAD fue aceptado por los clientes, mostrando la validez del trabajo realizado y mediante la cual se expresa la conformidad con los requisitos definidos. Además, donde queda plasmado que la herramienta se adapta perfectamente a las necesidades planteadas.

El acta de aceptación por parte del cliente se presenta a continuación:

UCI Universidad de las Ciencias Informáticas

Carta de conformidad de cliente con los requisitos.

La Habana, 29 de marzo de 2010

"Año del 52 Aniversario de la Revolución"

A: Universidad de las Ciencias Informáticas.

Mediante el presente documento se expresa la conformidad con los requisitos de la Herramienta CASE ABAD considerando que el trabajo realizado por los estudiantes se adapta perfectamente a las necesidades planteadas.

- El presente trabajo de diploma sirve de base a los futuros desarrolladores de la Herramienta, ya que les garantiza un mejor entendimiento de las funcionalidades del mismo, para posteriormente realizar las mejoras de la Herramienta con mayor facilidad.
- Los prototipos de interfaz no funcionales se adecuan completamente a las necesidades planteadas para la realización de la Herramienta.

De manera general los requisitos identificados constituyen una propuesta que integran muy bien las funcionalidades de las técnicas Análisis de Secuencia y Control de términos que formarán parte del módulo Técnicas en la Herramienta CASE que asiste a la arquitectura de información (ABAD). Consideramos que el trabajo realizado por los estudiantes es muy bueno, y se ha realizado con profesionalidad.

Y para que así conste se firma la presente a los 29 días del mes de marzo de 2010

Lic. Keyttia Pintón Almenares
Especialista en Información
Cliente Funcional

Ing. Alberto Tamayo Ramos
Jefe de Dpto. Universidad Digital

Dirección de Informatización

4.1.3 Métricas aplicadas en el Diagrama de Casos de Uso del Sistema

Para medir la calidad de la funcionalidad del diagrama de casos de uso del sistema, se utilizó un modelo de métricas orientado a objetos. Éste consta de cuatro atributos que se tuvieron en cuenta, los cuales son: Consistencia, Correctitud, Completitud y Complejidad.

Donde la Completitud consiste en el grado en que se han detallado los casos de uso más relevantes. La Correctitud consiste en el grado en que las interacciones entre actor y sistema soportan el modelo de negocio. La Complejidad viene dada por el grado de presentación de los elementos que describe el contexto y la claridad del sistema. Por otra parte, la Consistencia se basa en el grado en que los casos de uso del sistema describen las interacciones entre el usuario y el sistema.

A continuación se aplican un grupo de métricas al diagrama de casos de uso del sistema:

Primera Revisión:

Atributo	Factor	Métrica asociada	Valor
Completitud	Factor 1. ¿Han sido definidos todos los roles relevantes de usuario encargados de generar/ modificar o consultar información?	Métrica 1: Número de roles relevantes omitidos.	Número de roles relevantes omitidos: 0 Se presenta un 100%.
	Factor 2. ¿Se presenta una descripción resumida de todos los casos de uso?	Métrica 2. Número de casos de uso que no tiene descripción resumida.	Número de casos de uso que no tiene descripción resumida: 0 Se presenta un 100%.

	Factor 3. ¿Están definidos todos los requisitos que justifican la funcionalidad del caso de uso?	Métrica 3: Número de requisitos omitidos por caso de uso. Métrica 4: Número de casos de uso que tienen requisitos omitidos.	Número de requisitos omitidos por caso de uso: 2 Se presenta un 92%. Número de casos de uso que tienen requisitos omitidos: 0 Se presenta un 100%.
	Factor 4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en (crítico, secundario, auxiliar, opcional)?	Métrica 5. Número casos de que no han sido clasificados.	Número casos de que no han sido clasificados: 0 Se presenta un 96%.
			Se presenta un: 98.4%
Consistencia	Factor 5. ¿El nombre dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario?	Métrica 6: Número de casos de uso que tienen un nombre incorrecto.	Número de casos de uso que tienen un nombre incorrecto: 1 Se presenta un 87.5%.
	Factor 6. ¿Está adecuadamente redactado (en el lenguaje del usuario) el flujo de eventos?	Métrica 7: Grado de adecuación de la descripción del flujo de eventos para un caso de uso.	La descripción se define en el lenguaje del usuario. Se define el responsable de cada acción: 2. Se presenta un 75%.

	Factor 7. ¿La descripción del flujo de eventos se inicia con la descripción de una acción externa originada por un actor o por una condición interna del sistema claramente identificable?	Métrica 8: Número de casos de uso cuya descripción extendida no inicia con una acción externa o con una condición monitoreada por el sistema.	Número de casos de uso cuya descripción extendida no inicia con una acción externa o con una condición monitoreada por el sistema: 0 Se presenta un 100%.
	Factor 8. ¿Existe una adecuada separación entre el flujo básico de eventos y los flujos alternos y/o flujos subordinados?	Métrica 9: Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos.	Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos: 0 Se presenta un 100%.
			Se presenta un: 90.6%
Correctitud	Factor 9. ¿Representa el caso de uso requisitos comprensibles por el usuario?	Métrica 10: Número de casos de uso en que los requisitos representados no son comprensibles por el usuario.	Número de casos de uso en que los requisitos representados no son comprensibles por el usuario: 0 Se presenta un 100%.
	Factor 10. ¿Las interacciones definidas describen la funcionalidad requerida del sistema?	Métrica 11: Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del	Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema: 2 Se presenta un 75%.

		sistema.	
	Factor 11. ¿Se ajusta la representación del diagrama del caso de uso de acuerdo a lo normado en la metodología?	Métrica 12: Grado en que se ajusta el diagrama del caso de uso a la metodología	Grado en que se ajusta el diagrama del caso de uso a la metodología: 0 Se presenta un 100%.
	Factor 12. ¿Las interacciones definidas introducen mejoras al proceso actual?	Métrica 13: Número de casos de uso que deben ser modificados para mejorar el proceso actual.	Número de casos de uso que deben ser modificados para mejorar el proceso actual: 0 Se presenta un 100%.
			Se presenta un: 93.75%
Complejidad	Factor 13. ¿Los elementos dentro del diagrama están adecuadamente ubicados de manera que facilitan su interpretación?	Métrica 14: Número de elementos del diagrama que requieren reubicación.	Número de elementos del diagrama que requieren reubicación: 0 Se presenta un 100%.
			Se presenta un: 100%

Segunda Revisión:

Atributo	Factor	Métrica asociada	Valor
Completitud	Factor 1. ¿Han sido definidos todos los roles relevantes de usuario encargados de generar/ modificar o	Métrica 1: Número de roles relevantes omitidos.	Número de roles relevantes omitidos: 0 Se presenta un 100%.

	consultar información?		
	Factor 2. ¿Se presenta una descripción resumida de todos los casos de uso?	Métrica 2. Número de casos de uso que no tiene descripción resumida.	Número de casos de uso que no tiene descripción resumida: 0 Se presenta un 100%.
	Factor 3. ¿Están definidos todos los requisitos que justifican la funcionalidad del caso de uso?	Métrica 3: Número de requisitos omitidos por caso de uso. Métrica 4: Número de casos de uso que tienen requisitos omitidos.	Número de requisitos omitidos por caso de uso: 0 Se presenta un 100%. Número de casos de uso que tienen requisitos omitidos: 0 Se presenta un 100%.
	Factor 4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en (crítico, secundario, auxiliar, opcional)?	Métrica 5. Número casos de que no han sido clasificados.	Número casos de que no han sido clasificados: 0 Se presenta un 100%.
			Se presenta un: 100%
Consistencia	Factor 5. ¿El nombre dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario?	Métrica 6: Número de casos de uso que tienen un nombre incorrecto.	Número de casos de uso que tienen un nombre incorrecto: 0 Se presenta un 100%.

	Factor 6. ¿Está adecuadamente redactado (en el lenguaje del usuario) el flujo de eventos?	Métrica 7: Grado de adecuación de la descripción del flujo de eventos para un caso de uso	La descripción se define en el lenguaje del usuario. Se define el responsable de cada acción: 0 Se presenta un 100%.
	Factor 7. ¿La descripción del flujo de eventos se inicia con la descripción de una acción externa originada por un actor o por una condición interna del sistema claramente identificable?	Métrica 8: Número de casos de uso cuya descripción extendida no inicia con una acción externa o con una condición monitoreada por el sistema.	Número de casos de uso cuya descripción extendida no inicia con una acción externa o con una condición monitoreada por el sistema: 0 Se presenta un 100%.
	Factor 8. ¿Existe una adecuada separación entre el flujo básico de eventos y los flujos alternos y/o flujos subordinados?	Métrica 9: Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos.	Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos: 0 Se presenta un 100%.
			Se presenta un: 100%
Correctitud	Factor 9. ¿Representa el caso de uso requisitos comprensibles por el usuario?	Métrica 10: Número de casos de uso en que los requisitos representados no son comprensibles por el usuario.	Número de casos de uso en que los requisitos representados no son comprensibles por el usuario: 0 Se presenta un 100%.

	Factor 10. ¿Las interacciones definidas describen la funcionalidad requerida del sistema?	Métrica 11: Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema.	Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema: 0 Se presenta un 100%.
	Factor 11. ¿Se ajusta la representación del diagrama del caso de uso de acuerdo a lo normado en la metodología?	Métrica 12: Grado en que se ajusta el diagrama del caso de uso a la metodología	Grado en que se ajusta el diagrama del caso de uso a la metodología: 0 Se presenta un 100%.
	Factor 12. ¿Las interacciones definidas introducen mejoras al proceso actual?	Métrica 13: Número de casos de uso que deben ser modificados para mejorar el proceso actual.	Número de casos de uso que deben ser modificados para mejorar el proceso actual: 0 Se presenta un 100%.
			Se presenta un: 100%
Complejidad	Factor 13. ¿Los elementos dentro del diagrama están adecuadamente ubicados de manera que facilitan su interpretación?	Métrica 14: Número de elementos del diagrama que requieren reubicación.	Número de elementos del diagrama que requieren reubicación: 0 Se presenta un 100%.
			Se presenta un: 100%

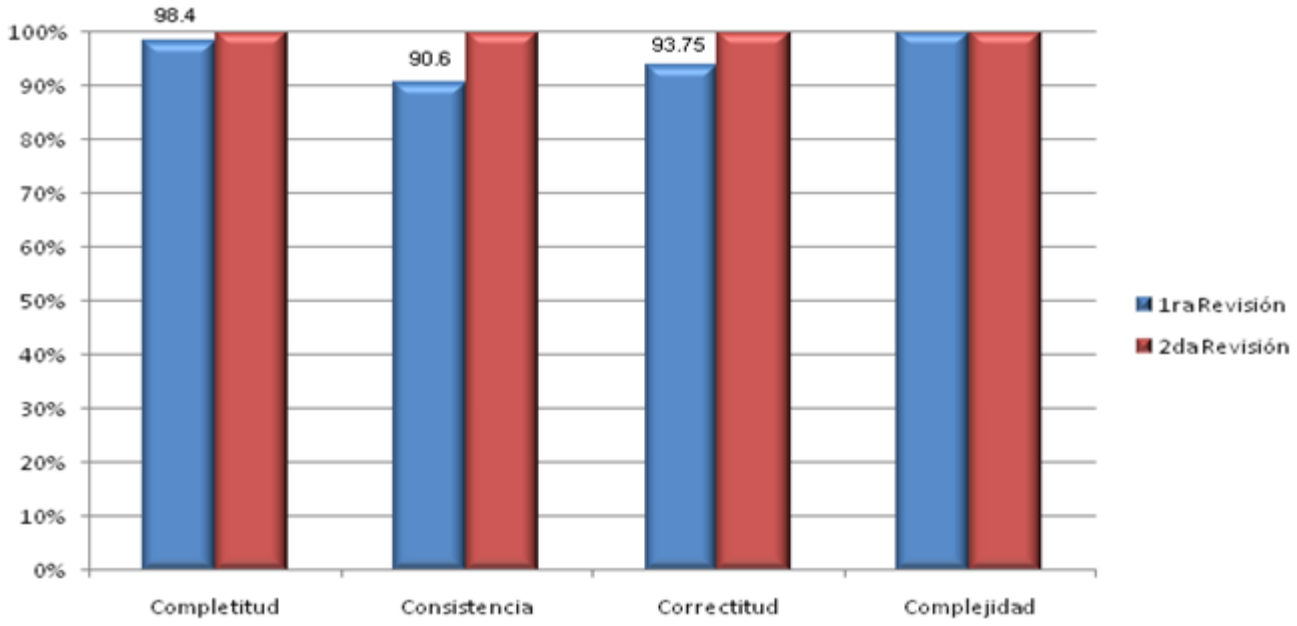


Figura 4: Gráfica de Factores de Métricas

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

Para comprobar el adecuado diseño de las clases y el nivel de reutilización de las mismas, se aplicó la métrica del TC. Se aplicará esta métrica para las principales clases de las dos capas definidas en el módulo.

Se presentó un **total de 7 clases** para un **promedio de atributos de 1.28** y **un promedio de operaciones de 4.28**. De esta forma, el umbral queda con los datos mostrados a continuación:

No	Nombre	Cantidad de Atributos	Cantidad de Operaciones	Tamaño
1	CC_GestionarTarjeta	0	3	Pequeño
2	CE_Tarjeta	3	3	Pequeño
3	CC_GestionarEjercicioAnalisisSecuencia	0	7	Pequeño
4	CE_Ejercicio	3	3	Pequeño
5	CE_Participante	3	3	Pequeño
6	CC_GestionarEjercicioControlTerminos	0	7	Pequeño

7	CC_GestionarParticipante	0	4	Pequeño

Clasificación	Cantidad Clases
Pequeño	7
Medio	0
Grande	0

Cantidad Clases	Promedio Atributos	Promedio Operaciones
7	1.28	4.28

Conclusiones

Después de realizados los flujos de trabajo de Negocio, Requisitos y Análisis y Diseño, aplicando las actividades y técnicas utilizadas en la Ingeniería de Requisitos, se logró captar las necesidades del cliente, identificando de esta manera los requisitos funcionales y no funcionales. Estos se detallaron a través de Casos de Uso y se validaron mediante métricas y Carta de Aceptación del Cliente, estas actividades arrojaron errores que fueron solucionados. En la aplicación de la métrica de la calidad de especificación de los requisitosse obtuvo primeramente un valor de aceptación de 0.90, ya en la segunda revisión el valor fue 0.97, en la aplicación de la métrica para el caso de uso todos los factores alcanzaron el 100% de aceptación luego de la segunda revisión y para la validación del diseño se aplicó la métrica tamaño de clase obteniéndose como promedio de atributos 1.28 y promedio de operaciones 4.28. Después de validados los resultados se puede decir que cuentan con la calidad requerida para cumplir las exigencias planteadas por el cliente.

Conclusiones Generales

Como resultado de haber cumplido el propósito del presente trabajo, luego de confeccionar un documento de apoyo a la comprensión de la situación problemática existente y su solución, a partir de la propuesta de diferentes artefactos que permitieron la correcta implementación del módulo Técnicas:

- Se realizó un estudio sobre las tendencias, tecnologías y metodologías más usadas en la actualidad para la construcción de sistemas informáticos similares y se seleccionó como la metodología idónea para cumplimentar los procesos de análisis y diseño, a RUP, conjuntamente con BPM, como apoyo al modelado del negocio, utilizando como herramienta Visual Paradigm para el modelado de todo el proceso.

- Se realizó la identificación de las funcionalidades que deberá poseer el producto, definiendo 18 requisitos funcionales y 13 no funcionales. Se realizó la descripción de su contexto productivo a partir de la identificación de 5 casos de uso, referenciados en sus descripciones por los requisitos funcionales definidos.

- Se construyó el modelo de diseño, lográndose una correcta comprensión del problema.

- Se modelaron los artefactos (diagramas de mapas de proceso, diagramas del proceso del negocio, descripciones textuales de casos de uso, diagrama de casos de uso del sistema, diagramas de clases del análisis, diagramas de colaboración, diagramas de secuencia y diagramas de clases persistentes del diseño) para guiar el proceso de implementación, exponiendo en cada caso el comportamiento y la interacción de los elementos que conforman cada entorno del software.

- Se comprobó la factibilidad del software a partir de la realización de la validación mediante métricas.

De esta forma, se puede afirmar que se alcanzó, satisfactoriamente el objetivo propuesto, creándose la modelación del módulo Técnicas.

Recomendaciones

- Realizar una segunda iteración donde se refinen los artefactos para lograr una mejor implementación.
- Realizar los artefactos restantes pertenecientes al diseño que son necesarios para continuar con el desarrollo del sistema.
- Profundizar en el estudio de la IR aplicando la etapa de administración de requisitos.
- Automatizar el proceso del trabajo de la audiencia con las tarjetas.

Bibliografía

1. **Delfino, E.***Integración de herramientas CASE usando Internet, CORBA y repositorios de Meta Información.* Universidad de la República de Uruguay, (Facultad de Ingeniería) : s.n., 2005.

2. **Brancheau, J. C y Wetherbe, J. C.***Information architectures: methods and practice. Information Processing & Management.* diciembre 1986. págs. 453 - 463. Vol. 22.

3. **Montes de Oca Sánchez de Bustamante, A.***Arquitectura de información y usabilidad: nociones básicas para los profesionales de la información.* 2004. Disponible en: http://bvs.sld.cu/revistas/aci/vol12_6_04/aci04604.htm .

4. **Garret, J. J.***The Elements of User Experience.* 2002. Disponible en: http://www.nosolousabilidad.com/articulos/historia_arquitectura_informacion.htm.

5. **Ronda León, Rodrigo.***Revisión de técnicas de Arquitectura de Información.* 5 de enero de 2007. Disponible en: http://www.nosolousabilidad.com/articulos/tecnicas_ai.htm.

6. **Ronda León, Rodrigo y Mesa Rábade, Y.***Análisis de secuencia: una herramienta para la Arquitectura de Información.* 2005. Disponible en: http://www.nosolousabilidad.com/articulos/analisis_secuencia.htm.

7. Disponible en: http://www.sap.com/solutions/sapbusinessobjects/large/intelligenceplatform/bi/dashboard-visualization/advanced-visualization/index.epx.

8. Disponible en: http://cmap.ihmc.us/conceptmap.html.

9. Disponible en: http://www.uoc.edu/symposia/spdece05/pdf/ID23.pdf.

10. Disponible en: http://www.axure.com.

11. **Barzanallana, Rafael.***Metodologías de desarrollo de software.* 2008. Disponible en: http://www.um.es/docencia/barzana.

12. **Pressman, Roger S.***Ingeniería del software. Un enfoque práctico.* 5. s.l. : McGraw-Hill/Interamericana de España. S.A, 2002.

13. **Jacobson, y otros.***El Proceso Unificado de Desarrollo de Software.* s.l. : Pearson Education, S.A, 2000.

14. **Burke, Eric y Coyner, Brian.**2000. Disponible en: http://oreilly.com/catalog/jextprockbk/chapter/ch04.pdf.

15. **Schwaber, K, Beedle, M y Martin, R. C.***Agile Software Development with SCRUM.* 2001. Disponible en: http://es-es.start2.mozilla.com/firefox?client=firefox-a&rls=org.mozilla:es-ES:official.

16. **Fuente, A y Lovelle, M.** 2006. Disponible en: https://forja.rediris.es/docman/view.php/227/369/Proceso%20Unificado.pdf.

17. *Rational Rose.* Disponible en: http://www.ciao.es/Rational_Rose_Enterprise_Edition__Opinion_612900.

18. *Visual Paradigm.* Disponible en: http://www.softpile.com/Development/Editors_and_IDEs/Review_19078_index.html .

19. **Saavedra, Jorge A y Gutierrez.** Santa Cruz : s.n., 8 de mayo, 2007. Ingeniero en sistema. Disponible en: 21. Saavedra, Jorge A, Gutierrez, Santa Cruz, Bolivia, (2007, mayo 8) Ingeniero en sistema Disponible en: http://jorgesaavedra.wordpress.com/category/patrones-grasp/.

20. **García, J.** 7 de mayo, 2005. Ingeniero de Software. Disponible en: http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

21. Ailonwebs. [En línea] http://ailonwebs.com/aplicaciones-web.php.

22. python. *python.org.* [En línea] http://www.python.org/about/.

23. **J, Soulie.** Cplusplus. [En línea] http://www.cplusplus.com/doc/tutorial.

24. Java. *Java.* [En línea] http://java.com/es/about/.

25. **Zukowski, John.***Java 2 J2SE 1.4 s.l.* Anaya Multimedia. 2003.

26. **Instituto Tecnológico de Hermosillo.** El lenguaje Java. [En línea] 12 de 2009. http://eddi.ith.mx/Curso/Contenido/java.htm.

27. **Jr. Frederick y P, Brooks.***No silver bullet: essence and accident of software engineering.* 1987.

28. **Ortas, I.***Determination of the extent of rhizosphere soil.* 1997.

29. **Christel, Michael G y Kang, Koy C.** [En línea] 2004. http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.012.html.

30. **Durán Toro, A.** [En línea] http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER00/toro.pdf.

31. **Reynoso, Carlos y Kiccillof, Nicolás.** Universidad de Buenos Aires : s.n., marzo 2004. Disponible en: http://www.willydev.net/descargas/prev/Estiloypatron.pdf.