Universidad de las Ciencias Informáticas

Facultad 10

Título: Análisis de un sistema automatizado a través de la técnica de Card Sorting u Ordenación de Tarjetas

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora: Yannelys Sola Padilla

Tutores: Lic. Marlen García Parrondo

Lic.Liudmila Amat Reyes

Co-tutor: Ing.Dayami Alfaro Montesino

Ciudad Habana, Julio de 2008
"Año 50 de la Revolución"

“Si el presente es de lucha, el futuro es nuestro”

Che

DECLARACIÓN DE AUTORÍA

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

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

Yannelys Sola Padilla

_____
Firma del Autor

Liudmila Amat Reyes

_____
Firma del Tutor

Marlen García Parrondo

_____
Firma del Tutor

DATOS DE CONTACTO

Lic. Liudmila Amat Reyes, Graduada en el curso 2006-2007 de Licenciatura en Estudios Socioculturales en el Instituto Superior Minero Metalúrgico de Moa con un índice académico de 5.21. Es profesora de la Universidad de las Ciencias Informáticas donde se encuentra impartiendo la asignatura de Filosofía y Sociedad. Durante su primer año de adiestramiento ha cursado 4 cursos de postgrados: Idolología Política; Docencia universitaria; Ciencia, tecnología y sociedad y Género y Minería. Ha participado en eventos de carácter nacional en internacional entre los que se encuentran: 1er Taller de Género y Minería Moa 2007, VII Congreso Iberoamericano de Ciencia, Tecnología y Género, FORUM de Ciencia y Técnica 2008.

Lic.Marlen García Parrondo, Graduada en junio del 2007 de Bibliotecología y Ciencias de la Información en la Universidad de La Habana. Actualmente es profesora de la Universidad de las Ciencias Informáticas donde se encuentra impartiendo la asignatura de Metodologia de la Investigación Científica y trabaja en la Infraestructura Productiva (IP) en el Dpto. de Arquitectura de Información. Ha realizado diferentes investigaciones como: Libros para la vida: un proyecto para la promoción de la lectura en la sala de pediatría del Instituto Nacional de Oncología y Radiobiología. Al igual que: La Gestión del Conocimiento en la Feria Internacional del Libro de La Habana. Ciudad Habana, 2005 y La Gestión del Conocimiento y su Medición en organizaciones cubanas de información en la actualidad, siendo este último la investigación realizada en el diploma de tesis. Llegando a publicar Libros para la vida http://www.bvs.sld.cu/revistas/aci/vol12_5_04/aci10504.htm y la investigación de la tesis en UCIENCIA 2007.

Ing. Dayami Alfaro Montesino, Profesora de Ingeniería de Software Facultad X, Universidad de Ciencias Informáticas. Teléfono: (53) (07) 835 8853. dalfaro@uci.cu. Ingeniera Informática, Universidad de las Ciencias Informática, 2007. Culminó sus estudios con índice académico de 4.76 puntos, obteniendo Título de Oro. Fue Alumna de Alto Aprovechamiento Académico y Alumna Ayudante desde el tercer año de su carrera, así como integrante de un proyecto productivo sobre la gestión bibliotecaria. Actualmente es analista principal de un proyecto productivo de exportación.

AGRADECIMIENTOS

A todas aquellas personas que durante la realización de esta investigación me han apoyado de una forma u otra cuando lo he necesitado.

En especial a mi mamá, abuela, mis papas, mis tías por darme fuerzas para seguir adelante porque son los que me han convertido en lo que hoy soy.

A toda mi familia por sus ánimos y esperanzas.

A mis compañeros de estudio, al equipo de trabajo del proyecto.

A mis amigos que siempre han compartido y han estado paso a paso conmigo esta etapa tan linda de la vida.

A mis tutoras: Marlen y Liudmila por su constante dedicación, ayuda y paciencia.

A la cotutora: Dayami por su gran ayuda, por encaminarme y apoyarme cuando más lo he necesitado.

A todos los profesores que a lo largo de la carrera han contribuido con mi formación como profesional.

Y a nuestro Comandante en Jefe, por concebir, entre sus grandes proyectos e ideas, este maravilloso sueño.

DEDICATORIA

Dedico este trabajo especialmente a mi mamá, mis abuelas, mis papás, mis tías, por su guía incondicional, ejemplo, dedicación, amor y porque han estado siempre escuchándome y apoyándome.

RESUMEN

En el presente trabajo se expone el Análisis de un Sistema Automatizado a través de la técnica del Card Sorting u Ordenación de Tarjetas como punto de partida para su posterior diseño e implementación. Este sistema permite gestionar toda la información referida a los procesos que se llevan a cabo para aplicar el mecanismo; permite una mayor efectividad cuando se pretende identificar y clasificar el contenido; responde a las dudas de los Arquitectos de Información de cómo agrupar el contenido con mayor efectividad, cuáles serán las necesidades de cada grupo y cuántas categorías usar; se puede delimitar y jerarquizar las problemáticas de mayor relevancia y frecuencia que inciden en el poco uso de la técnica; se utiliza la metodología Rational Unified Process (RUP) que hace uso del lenguaje de modelado UML; se obtiene como resultado el modelo de los flujos de trabajo: Negocio, Requerimientos y Análisis, los cuales constituyen la base para el Diseño, Implementación y posteriormente el Despliegue del sistema a automatizar.

PALABRAS CLAVE

Arquitectura de Información, Card Sorting, Sistema Automatizado.

INDICE

INTRODUCCIÓN

Durante las dos últimas décadas la arquitectura de la información se ha ganado un lugar en los planes de estudio de las carreras relacionadas con las Ciencias de la Información, así como en los eventos donde se abordan temas relacionados con el diseño de software**.** [1]

La Arquitectura de la Información no es más que el arte y la ciencia de organizar información para ayudar a las personas a cubrir sus necesidades de información eficientemente. Su objetivo principal es incrementar la capacidad de un sistema interactivo para que un usuario encuentre lo que necesita.

Para realizar esta organización centrada en el usuario, el arquitecto de información dispone de técnicas de ayuda en la toma de decisiones, como es la denominada "Card Sorting u Ordenación de Tarjetas".

La técnica de Card Sorting se basa en la observación de cómo los usuarios agrupan y asocian entre sí un número predeterminado de tarjetas etiquetadas con las diferentes categorías temáticas del sitio web. De esta forma, partiendo del comportamiento de los propios usuarios, es posible organizar y clasificar la información de un sitio web conforme a su modelo mental.[2]

La utilización de esta técnica posibilita explorar cómo las personas agrupan los conceptos e imaginan la organización de la información, mejorar la estructura de contenido dando sentido y amplitud a todas y cada una de las categorías que aparecen en la web, proporcionar datos concretos que pueden ser instanciados, además es barata, rápida e involucra a todo tipo de usuarios.

Desde finales de los 90 se han desarrollado varias herramientas informáticas para Card Sorting, fundamentalmente para facilitar el análisis estadístico de los resultados de los agrupamientos realizados por los usuarios, pero la mayoría de ellas aunque tengan una interfaz visual para trabajar

---

[1] Rodrigo Rondan León, 5 de enero del 2007. Revisión de técnicas de arquitectura de información. [Disponible en] : http://www.nosolousabilidad.com/articulos/tecnicas_ai.htm

[2] Hassan Montero, Y.; Martín Fernández, F.J.; Hassan Montero, D.; Martín Rodríguez, O. (2004). Arquitectura de la Información en los entornos virtuales de aprendizaje: Aplicación de la técnica de Card Sorting y análisis cuantitativo de los resultados. En: El Profesional de la Información, 2004, marzo-abril. [Disponible en]: http://www.nosolousabilidad.com/hassan/cardsorting.pdf

con las tarjetas, no cumplen con todos los ejercicios necesarios para desarrollar eficientemente la técnica.

En Cuba esta técnica se realiza manualmente. En realidad cuesta trabajo y es tedioso su desempeño debido a que hay que aplicársela a muchos usuarios para que el resultado sea el esperado, con la calidad requerida, y el producto web a diseñar tenga los requerimientos propuestos por el usuario y el cliente a la vez.

En Cuba la Arquitectura de Información desafortunadamente muchas veces no se presta atención, su aplicación y consolidación como término se ha explotado muy poco, en realidad un número cuantioso de organizaciones no la usan, por tanto menos emplean sus técnica a la hora de confeccionar un sitio Web. En el caso de la Universidad de las Ciencias Informáticas (UCI), como motor principal de la Revolución de Informatización, tiene como meta informatizar todos sus procesos que en ella se realizan y los que además, tienen que ver con otras esferas, para posteriormente generalizarlos al resto de las instituciones del país, los arquitectos de información para realizar la organización de categorías centrada en el usuario casi nunca utilizan técnicas de ayuda en la toma de decisiones, de ahí la **situación polémica:** Se necesita de un sistema automatizado que aplique el mecanismo de la técnica del Card Sorting u Ordenación de Tarjetas.

Una vez que se decide enfrentar este proyecto de investigación desde la producción interna, hay que afrontar los nuevos retos relativos a la solución óptima de la misma y luego de traducirse los elementos principales del negocio en términos de un sistema y modelar los elementos correspondientes a este flujo, entonces cabe preguntar: ¿cómo lograr tener una visión global esclarecedora del proyecto para los desarrolladores? De ahí se deriva el siguiente **problema de investigación:** ¿Cómo modelar en términos de clases, a partir de una ingeniería de requisitos, un sistema automatizado por medio de la técnica de Card Sorting u Ordenación de Tarjetas?
Para darle solución al problema se plantea el siguiente

Objetivo General: Modelar un sistema informático automatizado mediante la técnica del Card Sorting.

Objetivos específicos:

- Caracterizar el proceso de gestión de información mediante la técnica de Card Sorting u Ordenación de Tarjetas.
- Identificar el flujo de procesos de la automatización que asegura que los consumidores, usuarios y desarrolladores tengan un entendimiento común de la organización.
- Levantar requisitos para obtener las funcionalidades del sistema.
- Construir el Modelo del Sistema para obtener el modelo de diseño del producto.

El **objeto de estudio** está centrado en**:** La metodología para el desarrollo de una aplicación basada en Card Sorting, de ahí se puede delimitar su **campo de acción:** análisis de un sistema que realice Card Sorting. Para darle cumplimiento a los objetivos específicos se propone las siguientes **tareas de investigación:**

- Estudio del proceso de gestión de la información mediante la técnica del Card Sorting.
- Estudio de las tecnologías vinculadas a la gestión de la información existentes en la actualidad.
- Selección del proceso de desarrollo de software a utilizar.
- Selección de las herramientas idóneas que se utilizarán en el desarrollo del sistema informático.
- Descripción de los procesos que definen las operaciones propias de la técnica del Card Sorting.
- Realización del análisis del sistema utilizando el proceso de desarrollo de software seleccionado.

Para darle cumplimiento al problema se utiliza el **método histórico-lógico** que posibilita el estudio de la trayectoria y evolución de otras herramientas desarrolladas anteriormente y utilizarlas como punto de referencia y comparación de los resultados alcanzados. El **analítico-sintético** se emplea para realizar conclusiones a partir de la información que se procesa definiendo los rasgos característicos de la ingeniería de requisitos. Mediante la **modelación** se muestra la técnica del Card Sorting de una manera más sintetizada a como se muestra en la realidad, alcanzando un mejor entendimiento del procedimiento, permite la creación de propuestas o alternativas al sistema, descubrir nuevas relaciones y cualidades del objeto de estudio.

Al unísono de los métodos teóricos, y con el objetivo de obtener información del cliente y personas especializadas en el tema se aplica el **método empírico la entrevista** para definir la percepción directa del objeto, del problema de investigación, y obtener información concreta de las conversaciones que se tuvieron con los expertos del tema a cerca de los estudios relacionados con la técnica.

Estructura Capitular

En el Capitulo1 se hace referencia a la fundamentación teórica de la investigación a realizar. Incluye un estado del arte del tema tratado en el ámbito tanto nacional como internacional, de las tendencias, técnicas, tecnologías, metodologías y software existentes que de una forma u otra están relacionados con el tema que se aborda profundizando en ellos.

En el Capítulo 2 se describe el flujo de procesos para aplicar la técnica, los casos de usos, diagramas de actividades, modelo de objeto, se modela el negocio, se definen los actores y trabajadores del negocio, se especifican los requisitos funcionales y no funcionales del sistema, se describen los actores del mismo, el diagrama de casos de uso y la descripción de los casos de uso del sistema.

Durante el Capítulo 3 se modela los diagramas de clases del análisis, también se realiza un estudio de la factibilidad del desarrollo del sistema, utilizando como variante para la estimación el Análisis de Puntos de Casos de Uso y como conclusión se realizará una valoración sobre el resultado obtenido de la estimación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

Actualmente uno de los más grandes desafíos en los diseños de Sitios Web e Intranet es crear la Arquitectura de Información, este se hace mediante varias técnicas para una categorización de contenidos centrada en el usuario, siendo el Card Sorting una de éstas técnicas que proporciona una doble herramienta: como ayuda para la toma de decisiones en la etapa de diseño conceptual y para evaluar una organización concreta de categorías en etapas de evaluación de usabilidad.
En este capítulo se abordan las tecnologías, metodologías y software usados en el mundo, existentes que de una forma u otra están relacionados con el tema que se aborda, además de una visión general de las mismas.

1.1 ¿Qué es gestión?

Gestión: proceso mediante el cual se obtiene, despliega o utiliza una variedad de recursos básicos para apoyar los objetivos de la organización.

Figura 1 Funciones de la gestión.

1.1.1 Funciones de la gestión

Planificar: proceso de establecer objetivos con el fin de alcanzar determinados resultados.
(Establecimientos de objetivos, elaboración de planes, etc.)
Organizar: proceso de dividir el trabajo y de coordinar el logro de resultados que tienen un propósito común.

Dirigir: proceso de conducir y coordinar esfuerzos laborales de las personas que integran una organización. Función mediante la cual se ponen en marcha las tareas programadas.
Controlar: proceso de supervisar las actividades y resultados, comparándolos con los objetivos y tomando las acciones correctivas, si son necesarias. [3]

Dado los conceptos de las funciones de la gestión, se puede ver que estas están entrelazadas y que una acontece a la otra, se ve como en toda organización o proceso a desarrollar primero tenemos que planificar lo que vamos a ha hacer, luego organizamos las ideas o la información requerida, posteriormente se dirige los esfuerzos laborales o a la masa y luego controlamos todas las metas y planes que se trazaron para llevar a cabo su desempeño con la calidad requerida.

1.2 Gestión de la Información

Según Woodman[4] la Gestión de Información es todo lo relacionado con la obtención de información adecuada, en la forma correcta para la persona indicada, al costo adecuado en el tiempo oportuno, en el lugar apropiado para tomar la acción correcta.

Para Gloria Ponjuán[5] cuando se habla de gestión de información se refiere al proceso mediante el cual se obtienen, despliegan o utilizan recursos básicos (económicos, físicos, humanos, materiales) para manejar información dentro y para la sociedad a la que sirve. Tiene como elemento básico la gestión del ciclo de vida de este recurso y ocurre en cualquier organización. Es propia también de unidades especializadas que manejan este recurso en forma intensiva, llamadas unidades de información.

Por lo antes expuesto se puede concluir que la gestión de información es el proceso de analizar y utilizar la información que se ha recabado y registrado para permitir a los administradores de todos los niveles tomar decisiones documentadas, así como para mejorar los procesos, productos y servicios de la organización mostrando ser un creciente e importante indicativo de nuestra época y de nuestra cultura.

[3] Gestión de la información en las organizaciones. Principios, conceptos y aplicaciones. s.1.: Empresa grafica Haydee Santamaría, Palma Soriano (Versión Digital)
[4] Ibídem
[5] Ibídem

1.3 Características de la técnica Card Sorting

El Card Sorting no es más que una técnica para la adquisición de conocimientos basada en una aproximación constructivista que sirve para entender como los usuarios imaginan la organización de la información, explorar las formas en que las personas agrupan conceptos y terminologías y para entender al modelo mental de los usuarios proporcionando datos que pueden ser instanciados. De aquí que la técnica de Card Sorting se basa en la observación de cómo los usuarios agrupan y asocian entre sí un número predeterminado de tarjetas etiquetadas con las diferentes categorías temáticas de un futuro sitio web. De esta forma, partiendo del comportamiento de los propios usuarios, es posible organizar y clasificar la información de un sitio web conforme a su modelo mental, pudiéndose clasificar esta como una útil técnica para la categorización de contenidos centrada en el usuario.

Atendiendo a la clasificación de **Rosenfeld y Morville (2002)**, podemos diferenciar entre dos tipos de "Card Sorting": abierto y cerrado. En el "Card Sorting" abierto el usuario puede agrupar las categorías libremente en el número de conjuntos que crea necesario; mientras que en el cerrado, los grupos o conjuntos están predefinidos y etiquetados y el usuario únicamente deberá colocar cada categoría en el grupo que crea corresponda. Este segundo tipo de "Card Sorting" está recomendado para verificar si un diseño de información es familiar y comprensible para el usuario, mientras que el "abierto" tiene el objetivo de descubrir qué tipo de clasificación de categorías sería más correcto utilizar.[6]
Para proveer entonces un mejor alcance del mecanismo de Card Sorting a continuación se muestra una serie de herramientas informáticas que aplican esta técnica.

1.4 Software que aplican la técnica de Card Sorting el mundo

Actualmente existen en el mundo varios software que aplican la técnica del Card Sorting utilizadas por los arquitectos de información en la toma de decisiones para confeccionar un sitio web, seguidamente se explicaran cada una de ellas.

CardZort

CardZort es una aplicación informática que funciona la tarjeta de ejercicios de clasificación. Su objetivo principal es ofrecer un completo asistido por ordenador sistema que permite la rápida creación y ejecución de ejercicios de clasificación de tarjetas, y el análisis de los grupos resultantes a través de

[6] Op. Cit. (2)

análisis de agrupamiento. Esta aplicación hace uso de una metáfora gráfica que imita de cerca el verdadero proceso de selección de tarjeta.

CardCluster

CardCluster es otra aplicación que hace el grupo de análisis sobre los datos obtenidos a partir de CardZort y hace análisis de etiqueta para visualizar las preferencias de los usuarios sobre los posibles nombres de los grupos resultantes.

EZSort

EZSort es un software desarrollado por IBM, pero desde hace tiempo ya no es actualizado ni se puede descargar desde el sitio web de IBM.

CardSword

CardSword es una herramienta open source que permite llevar a cabo pruebas de Card Sorting, así como el análisis cuantitativo de los resultados. Permite realizar estudios del Card Sorting de forma virtual, y además puede realizar dos tipos de análisis cuantitativos de los resultados: clústering (dendograma) y redes. Actualmente se encuentra en su versión Beta 0.9, pero todavía quedan bastantes funcionalidades por implementar.[7]

WEBCAT

WEBCAT es un producto interno de ALBA Software[8] que se utiliza como plataforma base para el desarrollo de la mayoría de las aplicaciones Web que desarrollamos. Entre estas aplicaciones se encuentra Gestorweb aplicación modular diseñada para llevar los procesos de negocio de la empresa en los que intervienen usuarios externos a una plataforma Web.

[7] Hassan Montero, Y. 6 Marzo 2005. Card Sword: Organización de categorías centrada en el usuario. [Disponible en]: http://www.nosolousabilidad.com/hassan/curso_madrid.htm

[8] ALBA Software provee de la tecnología y el soporte necesarios para que cualquier empresa se administre todos sus servicios Internet, obteniendo así un mayor control, seguridad, integración con su sistema informático, y autonomía para la implementación de nuevos servicios.

En estos momentos WEBCAT no se mantiene fuera de los productos de ALBA Software pero no se descarta dar soporte en un futuro como producto autónomo.

WEBCAT simplifica y acelera el desarrollo de aplicaciones Web sobre la arquitectura J2SE (Java 2 Estandar Edition) propuesta por Sun microsystems y adoptada por la mayoría de los grandes fabricantes de software (Oracle, Bea, Borland, Netscape, IBM, etc.). Este no está enfocado a ningún sector específico. Con WEBCAT se han desarrollado catálogos Web en el sector del plástico, automatización de fuerza de ventas en el sector de marroquinería, un interface Web para un servidor de Fax. Cualquier aplicación que quiera llevarse a la Web puede beneficiarse de la arquitectura que propone WEBCAT ahorrando tiempos de desarrollo y recursos.[9]

WebSort

Es un instrumento de software a base de web que permite a investigadores realizar estudios de clase de tarjeta remotos.

Web Sort no es gratis, teniendo una versión demo con un límite de participantes por prueba de 10. De todas formas Web Sort permite exportar los resultados para ser analizados con EZCalc, por lo que siempre queda la opción de crear la misma prueba para cada diez participantes y después realizar el análisis conjunto con EZCalc. [10]

CardSort

El card sort se trata de qué miran primero los usuarios. Es una buena herramienta cuando los usuarios nunca han visitado el sitio o cuando se ha rediseñado profundamente.

El objetivo de este test es ver si la arquitectura del sitio tiene sentido para el usuario. Una regla es que incluya categorías amplias de alto nivel de información. Cada encabezamiento debe tener un enlace a la página de inicio del sitio.

Una vez examinada estas herramientas, quedaría analizar el desarrollo de la técnica del Card Sorting en nuestro país, siendo esta un punto crucial en la investigación.

---

[9] WEBCAT. Consultado 20 febrero del 2008. [Disponible en:] http://www.gestorweb.com/docu/webcat_intro.html

[10] Hassan Montero, Yussef. 1 de junio del 2007. Materiales del curso "Arquitectura de Información para la Web (3ª Edición)". http://www.bitacoras.sidar.org/g4/index.php?2005/03/06/5-card-sword-organizacion-de-categorias-centrada-en-el-usuario-ejemplo-de-prueba-con-cardsword

1.5 La técnica de Card Sorting en Cuba

En Cuba respecto al Card Sorting no se ha implementado ninguna aplicación que desarrolle el ejercicio. En nuestra Universidad más bien se usa muy poco. Dentro de las causas que hacen posible que esta no se use están: las características propias de cada proyecto, no siempre resulta factible hacerla, ya sea porque no se cuenta con las condiciones necesarias (tiempo e infraestructura fundamentalmente) o porque son más fáciles de ajustar otras técnicas (análisis de homólogos, observación participante, entrevistas, tormentas de ideas…); el arquitecto de información no domina bien la técnica y prefiere aplicar otras; no se realiza el inventario previo sin el cual no es posible ni efectivo aplicar la técnica porque no se conocen todas las entidades de recursos de información con las que se va a interactuar y aunque existen herramientas informáticas que ayudan a aplicar esta técnica no siempre son fáciles de usar o de libre acceso resultando engorroso aplicar un algoritmo a base de comparaciones y estadísticas de manera no automatizada como es este.

Con un mayor uso de esta técnica en la UCI se obtendrían arquitecturas de información más profesionales y mucho más intuitivas para el usuario garantizando una menor cantidad de iteraciones para lograr la aceptación final de la arquitectura de información por parte de los clientes. También el resultado final estuviese más acorde a las necesidades de la audiencia del producto, esto se logra diseñando un producto usable, accesible y sobre todo, centrado en el usuario y la técnica evidentemente proporciona todo esto.

Por lo anteriormente dicho, es que se necesita implementar un software que automatice el mecanismo, pues se lograría un diseño centrado en el usuario optimizando el tiempo de aplicación de la técnica así como el aprendizaje de la misma por parte de los arquitectos de información y los recursos a emplear. Se automatiza el proceso, por tanto, no es necesario gastar recursos elaborando tarjetas cada vez que sea necesario aplicar la técnica. Se pueden compartir y reutilizar recursos a partir de la aplicación de la técnica en otros proyectos. Facilita la ubicuidad tanto de expertos (arquitectos de información) como de los usuarios, aunque siempre con estos últimos es conveniente combinar el Card Sorting con otras técnicas como la entrevista y la observación participante. Se puede incorporar el aprendizaje del uso de la herramienta como un objetivo del curso de arquitectura de información que se imparte hoy en la UCI. Puede formar parte de algún proyecto de I+D para la temática de Arquitectura de Información de manera que se pueda ir perfeccionando a partir de experiencias prácticas e investigaciones que se realicen. Puede ser comercializable, pues no se tiene conocimiento de que existan muchas herramientas similares en el mundo.

Es válido mencionar en este punto la importancia que posee la presencia de las metodologías de trabajo en el desarrollo de las actividades concernientes a la ejecución del Card Sorting, pues sin su desempeño, todo lo antes expuesto no sería posible, lo cual se sustenta seguidamente.

1.6. Metodologías de trabajo

La calidad en el desarrollo y mantenimiento del software se ha convertido hoy en día en uno de los principales objetivos estratégicos de las organizaciones. Para obtener un producto de calidad se hace necesario un estudio de las diferentes tecnologías y de esta forma decidir la adecuada, teniendo en cuenta un aspecto imprescindible, la seguridad del sistema.

Para garantizar la calidad y efectividad de un software, es necesario determinar que metodología a usar, a veces no se sabe escoger la más correcta pero cuando los proyectos son de envergadura sí se debe tener en cuenta una buena elección.

Las propuestas de metodologías más tradicionales se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán pueden incluso limitar la capacidad de los desarrolladores para llevar a cabo el proyecto, sin embargo las metodologías ágiles forman parte del movimiento de desarrollo ágil de software, que se basan en la adaptabilidad de cualquier cambio como medio para aumentar las posibilidades de éxito de un proyecto.

En la siguiente tabla se muestran las diferencias entre estos distintos procesos de desarrollo.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo de desarrollo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Tabla 1 Diferencias entre la Metodología Tradicional y Ágil

Para dar una idea de que metodología se va a utilizar y cual se adapta más a nuestro medio, se mencionará tres de ellas de las que se considera las más importantes, por ejemplo: RUP, XP y SCRUM. A continuación se explican las características de cada una de ellas.

1.6.1 Metodologías Tradicionales

1.6.1.1 RUP (Rational Unified Process)

El Proceso Unificado fue desarrollado por Ivar Jacobson y Rumbaugh[11], y otros de la Rational como el proceso complementario al UML. El RUP es un proceso de ingeniería de software, bien definido y

[11] Jacobson, I., Booch y G, Rumbaugh J. El Proceso Unificado de Desarrollo de Software, 2000 Addison Wesley (Versión Digital)

estructurado; al mismo tiempo es un proceso adaptable a las necesidades y características de cada proyecto específico y puede acomodar una gran variedad de procesos fundamentales:

- Utiliza el paradigma de orientación a objetos para su descripción.
- Es un marco de proceso configurable para satisfacer necesidades específicas.
- Implementa las mejores prácticas de desarrollo de software.

Características principales de RUP

- Dirigido por casos de uso

Los casos de uso capturan requerimientos funcionales y representan piezas de funcionalidad que brindan un resultado de valor al usuario.

Figura N º 2

- Centrado en una arquitectura

Comprende los aspectos estáticos y dinámicos más importantes del sistema.

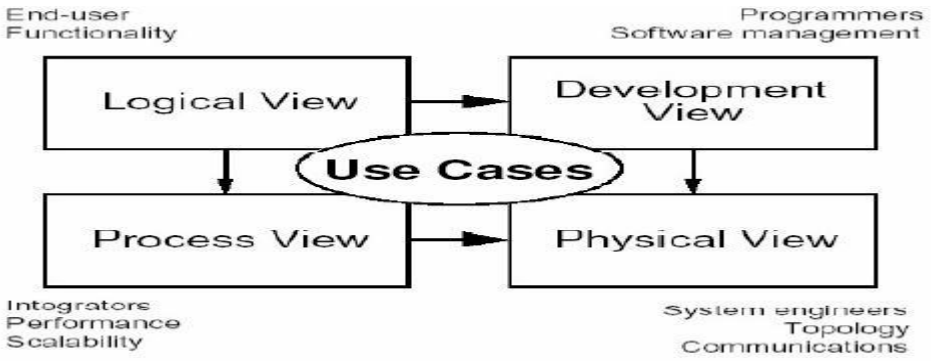


Figura Nº 3

- Iterativo e incremental

El trabajo se divide en piezas pequeñas o mini proyectos; cada uno proveyendo un subproducto incremental.

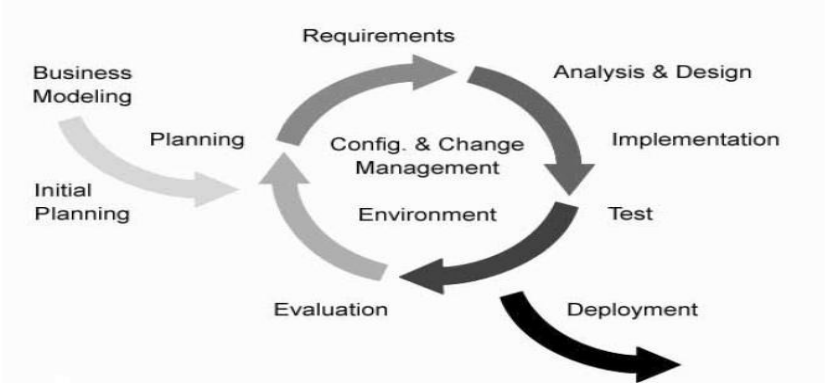


Figura Nº 4

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

- Inicio, El Objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración, En esta etapa el objetivo es determinar la arquitectura óptima.
- Construcción, En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- Transmisión, El objetivo es llegar a obtener el reléase del proyecto.

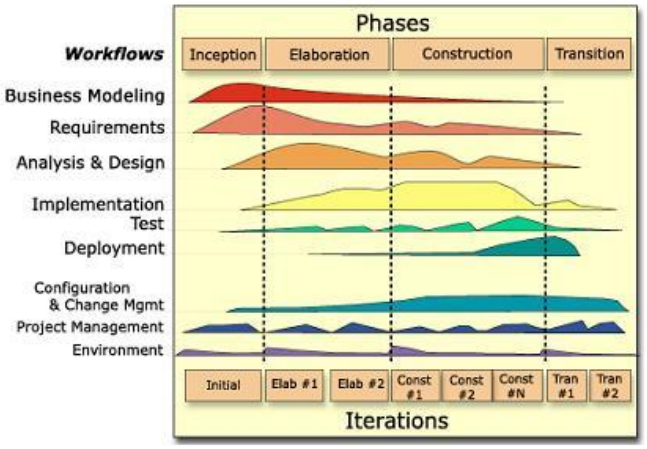


Figura Nº 5

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.

Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

Disciplina de Desarrollo

- Ingeniería de Negocios: Entendiendo las necesidades del negocio.
- Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado esta presente.

Disciplina de Soporte

- Configuración y administración del cambio: Guardando todas las versiones del proyecto.
- Administrando el proyecto: Administrando horarios y recursos.
- Ambiente: Administrando el ambiente de desarrollo.
- Distribución: Hacer todo lo necesario para la salida del proyecto

Los elementos del RUP son:

- Actividades, Son los procesos que se llegan a determinar en cada iteración.
- Trabajadores, Vienen hacer las personas o entes involucrados en cada proceso.
- Artefactos, Un artefacto puede ser un documento, un modelo, o un elemento de modelo.[12]

1.6.2 Metodologías Ágiles

1.6.2.1 eXtreme Programming (XP)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizadas para proyectos de corto plazo, equipo y cuyo plazo de entrega era ayer. 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.

[12] Pressman, R. S. (2005). Ingeniería del Software: Un enfoque Práctico. La Habana

Las características fundamentales son:[13]

- Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.
- Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

Las características esenciales de XP están organizadas en 3 apartados: historias de usuario, roles, proceso y prácticas. Los cuales permiten de manera eficiente desarrollar cualquier software de manera ágil.

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Respecto a los roles XP existen diversas fuentes que abordan al respecto, aquí se hará referencia a los roles según el creador de la metodología. Los principales roles descritos por Beck son:

Programador: Escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

Cliente: Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.

Encargado de Pruebas (Tester): Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Encargado de seguimiento (Tracker): Proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo

[13] Mendoza Sánchez, María A. Junio 7 del 2004. Metodologías De Desarrollo De Software.[Disponible en]: http://www.willydev.net/InsiteCreation/v1.0/Descargas/cualmetodologia.pdf

real dedicado, comunicando los resultados para mejorar futuras estimaciones.

Entrenador (Coach*)*: Es responsable del proceso global.

Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto.

Gestor (Big boss*)*: Es el vínculo entre clientes y programadores. Su labor esencial es de coordinación.

¿Qué es lo que propone XP?

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

Derechos del Cliente

- Decidir que se implementa
- Saber el estado real y el progreso del proyecto
- Añadir, cambiar o quitar requerimientos en cualquier momento
- Obtener lo máximo de cada semana de trabajo
- Obtener un sistema funcionando cada 3 o 4 meses

Derechos del Desarrollador

- Decidir como se implementan los procesos
- Crear el sistema con la mejor calidad posible
- Pedir al cliente en cualquier momento aclaraciones de los requerimientos
- Estimar el esfuerzo para implementar el sistema
- Cambiar los requerimientos en base a nuevos descubrimientos

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del Sistema.

- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales. [14]

1.6.2.2 SCRUM

Scrum es una metodología para la gestión de proyectos, expuesta por Hirotaka Takeuchi e Ikujiro Nonaka [15] en el que ponen de manifiesto que:

- El mercado competitivo de los productos tecnológicos, además de los conceptos básicos de calidad, coste y diferenciación, exige también rapidez y flexibilidad.
- Los nuevos productos representan cada vez un porcentaje más importante en el volumen de negocio de las empresas.
- El mercado exige ciclos de desarrollo más cortos.

Los creadores de esta metodología para llevar a cabo su desempeño compararon el desarrollo tradicional de ciclo de vida formado por fases separadas y equipos especializados con las carreras de relevos simulándolo con el juego del rugby, y de él se toma el término scrum.

Nonaka y Takeuchi extraen las bases de scrum de las prácticas que observan en las empresas con buenos resultados de rapidez y flexibilidad en la producción: Xerox, Canon, Honda, NEC, Epson, Brother, 3M o Hewlett-Packard.

Aunque surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados sistemas de software.
En el desarrollo de software scrum está considerado como modelo ágil por la Alianza Ágil. El scrum es simple en su composición, se puede ver como 3 grandes bloques principales:

- Las personas
- Las ceremonias
- Los documentos o artefactos

[14] Canós, José H, Letelier Patricio y Penadés Mª Carmen. Metodologías Ágiles en el Desarrollo de Software.[Disponible en]: http://www.willydev.net/descargas/prev/TodoAgil.Pdf

[15] Takeuchi, Hirotaka ; Nonaka, Ikujiro The New Product Development Game[Harvard Business Review Ene-Feb 1986]

Ningún bloque es más importante que el otro, cada uno aporta un valor equivalente y constituyen entre si un esquema de desarrollo sostenido y sólido.

Scrum es un proceso ágil que sirve para administrar y controlar el desarrollo de software. El desarrollo se realiza en forma iterativa e incremental. Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad. Las iteraciones en general tienen una duración entre dos y cuatro semanas. Está diseñado especialmente para adaptarse a los cambios en los requisitos, estos se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares, de esta forma se puede adaptar en tiempo real el producto que se está construyendo a las necesidades del cliente.

Después de haber explicado las metodologías agiles y tradicionales se puede decir que no existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto, es decir recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema, etc. Históricamente, las metodologías tradicionales han intentado abordar la mayor cantidad de situaciones de contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes; sin embargo las metodologías ágiles por su sencillez , tanto en su aprendizaje como en aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo, ofrecen una solución casi a medida para una gran cantidad de proyectos que tienen estas características. A continuación se explicara que metodología de trabajo se determino a utilizar para el desarrollo del sistema a automatizar.

¿Por qué usar RUP?

La metodología de desarrollo a utilizar es la de Rational Unified Process (RUP) debido a que es una de las metodologías de gestión de proyectos más innovadoras que se utiliza en la Universidad de las Ciencias Informáticas, esta 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 requerimientos no se pueden definir completamente al principio, permite evaluar tempranamente los riesgos en lugar de descubrir problemas en la integración final del sistema, reduce el costo del riesgo a los costos de un solo incremento, acelera el ritmo del esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan para obtener resultados claros a corto plazo, 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.

1.7 Lenguaje de Modelado (UML)

RUP utiliza UML (por sus siglas en inglés, Unified Modeling Language) pues es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

- Diagramas de estructura estática: Describen las propiedades estructurales del sistema.
    - Diagrama de clases: Conjunto de clases, interfaces y colaboraciones; así como sus colaboraciones.
    - Diagrama de objetos: Conjunto de objetos y sus relaciones.
    - Diagrama de casos de uso: Conjunto de casos de uso y actores y sus relaciones.
- Diagramas de comportamiento:
    - Diagramas de interacción (secuencia y colaboración): Objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos.
    - Diagrama de estados: Muestra una máquina de estado que consta de estados, transiciones, eventos y actividades.

- Diagrama de actividad: Es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.

• Diagramas de implementación:
  - Diagrama de componentes: Organización y las dependencias entre un conjunto de componentes.
  - Diagrama de despliegue: Configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

1.8 Herramientas de Modelado

Cuando se habla de herramientas de modelado en la disciplina de Ingeniería del Software, es importante mencionar las herramientas CASE. Estas "están tomando cada vez más relevancia en la planeación y ejecución de proyectos que involucren sistemas de información, pues suelen inducir a sus usuarios a la correcta utilización de metodologías que le ayuden a llegar con facilidad a los productos de software construidos".[16]

La herramienta CASE (Computer-Aided Systems Engineering) cuyo significado en español es ingeniería de sistemas asistida por ordenador, es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo de sistemas. Con el objetivo de automatizar o apoyar una o mas fases del ciclo de vida del desarrollo de sistemas. Actualmente la oferta de herramientas CASE es muy amplia y tenemos por ejemplo el Visual Paradigm, el Umbrello y el Rational Rose, a continuación se mencionará las características de ellas.

1.8.1 Visual Paradigm- UML

Visual Paradigm para UML es una herramienta CASE, considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Fue creada para el ciclo vital complete del desarrollo del software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación.

Visual Paradigm-UML también proporciona características tales como generación del código, ingeniería inversa y generación de informes. Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de clases. Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Está diseñada para usuarios interesados en

[16] Quintero, Juan Bernardo, y otros. 2005. Un estudio comparativo de herramientas para el modelado con UML. Colombia: s.n., 2005.

sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros. [17]

1.8.2 Rational Rose Enterprise

Rational Rose Enterprise es una herramienta que posee la capacidad de ver, crear, modificar y manipular los componentes de las notaciones UML. Esta cubre los nueve flujos de trabajo de la metodología RUP (Rational Unified Process) para el desarrollo de proyectos de software, seis flujos de ingeniería y tres de soporte. Rose es la herramienta de Rational para la etapa de análisis y diseño de sistemas. Además monitorea el tiempo de desarrollo, ayuda en la comprensión del entorno del sistema y la comunicación entre los miembros del equipo. Dentro de las características fundamentales están:

- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++.
- Característica de control por separado de componentes modelo que permite una administración más factible y el uso de modelos.
- Soporte de ingeniería Forward y/o reversa para algunos de los conceptos más comunes de Java 1.5.
- La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables.
- Soporte Enterprise Java Beans™ 2.0.
- Capacidad de análisis de calidad de código.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- Capacidad de crear definiciones de tipo de documento XML para el uso en la aplicación.
- Integración con otras herramientas de desarrollo de Rational.
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo

[17] Visual Paradigm International. Visual Paradigm.10 Reasons to Choose Visual Paradigm. [En línea] [Disponible en]: http://www.visual-paradigm.com/aboutus/10reasons.jsp.

1.8.3 Umbrello

Umbrello es una herramienta CASE libre para crear y editar diagramas UML, que ayuda en el proceso del desarrollo de software, facilitará la creación de un producto de alta calidad, especialmente durante fases de análisis y diseño del proyecto. Genera código automáticamente en los lenguajes C++, Java, PHP, entre otros, y se puede importar de C++. Presenta varias opciones para el manejo de los diagramas en el momento de su visualización, exportación o impresión. [18]

¿Por qué usar Visual Paradigm?

La herramienta Case que se decidió emplear es el Visual Paradigm , dado que es una herramienta que la universidad ha incrementado los niveles de aceptación por su robustez, usabilidad y portabilidad, además es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; permite control de versiones; facilita la organización, visualización, diseño, integración y despliegue mediante diagramas; ayuda al equipo de desarrollo a mejorar la construcción del modelo del proceso de desarrollo de software, maximizando y acelerando la producción del equipo y las contribuciones individuales. También cuenta con una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software, lo cual garantiza la calidad del producto final.

1.9 Entorno Integrado de Desarrollo (IDE)

1.9.1 Visual Studio

Es uno de los IDE (Integrated Development Environment, Entorno integrado de desarrollo) más populares por su rendimiento en cualquier plataforma, facilitando de manera increíble el desarrollo de .NET. Su atractivo reside en el hecho de que con él podemos incrementar espectacularmente nuestra productividad en poco tiempo. Escrito tanto para desarrolladores de .NET principiantes como experimentados, Visual Studio.

Con Visual Studio los desarrolladores pueden:

- Crear aplicaciones de línea de negocio usando Visual Basic, C#, C++ y J#
- Construir aplicaciones para Windows, la Web y dispositivos móviles todo desde el mismo entorno unificado de desarrollo

[18] Environment, K Desktop. 2008. K Desktop Environment. [En línea]. [Disponible en]: http://docs.kde.org/stable/es/kdesdk/umbrello/introduction.html.

- Desarrollar aplicaciones cliente/servidor usando servicios Web y herramientas integradas de diseño para acceder a datos remotos

Con las Ediciones Visual Studio, los usuarios pueden:

- Aprender a programar utilizando un entorno de desarrollo sencillo y directo con tutoriales integrados
- Evaluar el .NET Framework para el desarrollo para Windows y la Web
- Crear aplicaciones interesantes y divertidas para disfrute personal o para compartir con amigos

1.10 Herramientas de Desarrollo

C#

Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma.NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes (más notablemente de Delphi y

C#, es actualmente uno de los lenguajes de programación más populares en informática y comunicaciones. Con el objetivo de permitirles a los programadores abordar el desarrollo de aplicaciones complejas con facilidad y rapidez.

Dada la sugerencia del cliente de todos los lenguajes que se pueden desarrollar en la herramienta Visual Studio se escogió el C# por que es un lenguaje que resulta mas común entre los desarrolladores del proyecto, es actual. C# incorpora en el propio lenguaje elementos que se han demostrado ser muy útiles, soporta todas las características propias del paradigma de la programación orientada a objetos: encapsulación, herencia y polimorfismo. Delega la gestión de memoria. No pueden usarse variables que no hayan sido inicializadas. Sólo se admiten conversiones entre tipos compatibles. Además el software es robusto, duradero y económico.

XML

XML (Lenguaje de Marcas eXtensible) consiste en una serie de reglas, pautas para planificar formatos texto de manera que produzcan archivos que sean fácilmente generados y leídos (por un ordenador)

que son inequívocos, y que evitan dificultades comunes como la falta de extensibilidad, falta de soporte para la internacionalización o localismo, y la dependencia de una determinada plataforma.

Principales características

- Es una arquitectura más abierta y extensible. No se necesita versiones para que puedan funcionar en futuros navegadores. Los identificadores pueden crearse de manera simple y ser adaptados en el acto en internet/intranet por medio de un validador de documentos (parser).
- Mayor consistencia, homogeneidad y amplitud de los identificadores descriptivos del documento con XML
- Se podrá hacer el intercambio de documentos entre las aplicaciones tanto en el propio PC como en una red local o extensa.

Conclusiones

En este capitulo se realizó un estudio acerca del estado del arte del tema tratado a diferentes niveles, llegando a la conclusión de la necesidad de desarrollar un sistema automatizado centrado en el usuario mediante la técnica del Card Sorting, dado por sus características, que no permiten la adaptación de otro sistema de los ya puestos en el mercado. Además se hizo un estudio de las diferentes metodologías y tecnologías tendientes en el mundo actual lleno de nuevos avances, a la hora de desarrollar este tipo de aplicaciones. La decisión tomada estuvo basada en la política de uso de herramientas con soporte multiplataforma y licencias de utilización libre, teniendo en cuenta las restricciones existentes en nuestro país, para el desarrollo de software debido al bloqueo y las exigencias de nuestros clientes. Para cumplir con dichas condiciones las herramientas y tecnologías utilizadas para el desarrollo fueron: Metodología RUP con notación UML para el desarrollo de la documentación, herramienta CASE con modelado UML: Visual Paradigm y para continuar el proceso de implementación del software se utilizarán herramientas de desarrollo como C# y XML.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Introducción

En el presente capítulo se describe el flujo de procesos para aplicar la técnica, el objeto de estudio. Se presenta la propuesta del sistema a implementar. Se define el modelo del negocio derivándose de estos últimos los actores, trabajadores, casos de usos y diagrama de actividades, modelo de objetos; las especificaciones de los requisitos de software, el diagrama de casos de uso del sistema; la descripción de estos y la modelación de los diagramas de clases del análisis correspondiente; donde se aprecia la interacción del usuario con las diferentes interfaces del sistema y las clases entidades involucradas.

2.1 ¿Flujo actual de los procesos para aplicar la técnica?

Para la realización de la gestión de información centrada en el usuario mediante la técnica del Card Sorting, se tiene que realizar una sucesión de procedimientos entre los cuales se pueden señalar: determinar el contenido a tratar en el sitio que se va diseñar; el proceso de crear las tarjetas; seleccionar los participantes (Audiencia); seleccionar las secciones de ordenación y almacenar los resultados, seguidamente se explica cada uno de los procesos.

- Determinar la lista de tópicos (contenido)

Cada tópico no debe ser ni muy genérico ni muy específico. Debe representar una parte del contenido o de la funcionalidad que necesita organizarse. La longitud ha de ser manejable. Siempre que sea posible, deben evitarse las pistas que conduzcan a los usuarios a organizar los tópicos de una forma determinada. También deben evitarse tópicos que incluyan “términos de agrupación” (Archivo, Edición, FAQs,..). Han de tener significado para los participantes en la sesión.

- Crear las tarjetas

Cada tópico se escribe en una tarjeta. Esto es lo más delicado del proceso porque cuando el etiquetado no es claro los usuarios no pueden agrupar bien elementos que no comprenden, por eso en determinadas ocasiones es preciso una pequeña descripción, que debe ser legible y tener tarjetas de sobras por si los usuarios necesitan crear agrupaciones.

¿Qué escribir en una tarjeta?

Las tarjetas no son categorías intermedias, es decir, no contienen ninguna otra categoría, son elementos finales que no se pueden agrupar más.

La tarjeta pueden ser categorías de último nivel cuando los elementos que contiene son iguales y no tiene sentido crear una tarjeta para cada uno. En otros casos las tarjetas son también elementos concretos y únicos.

Todo dependerá del contenido y lo lejos que se esté interesado en llegar. Por ejemplo, el negocio a tratar es el diseño de un sitio web y se tienen muchos con diferentes prestaciones y precios, sería de interés averiguar cómo los clasifica la gente, si por precio, prestaciones, accesibilidad, etc. Para ello incluyendo un número representativo de modelos concretos en las tarjetas, se puede observar como los usuarios las agrupan.

Número de tarjetas

El número máximo de tarjetas puede ser alrededor de 50. Con más de 50 la prueba es demasiado larga, los participantes se cansan y las categorías creadas son de peor calidad.

A mayor número de tarjetas, la calidad de las categorías será menor y más participantes serán necesarios.

Tarjetas con nombres problemáticos

Cuando hay productos con nombres excesivamente extraños, que no tienen sentido sin una explicación, el problema es grave. El usuario no puede agrupar cuando no sabe lo que está agrupando.

Ante este problema se tienen dos opciones:

- Se puede aprovechar el test para demostrar que los nombres de algunos productos son problemáticos, nadie los entiende, nadie los agrupa bien y por tanto nadie los encuentra en el sitio. La prueba será útil para esto, pero al existir tarjetas malas las categorías resultantes no serán tan buenas. En ese caso debería repetirse el test una vez que se cambie el nombre a esas categorías problemáticas.
- Si únicamente se quiere descubrir las mejores categorías posibles sin cambiar el nombre de los elementos, podemos añadir una breve explicación de 3 o 4 palabras, al nombre de la tarjeta para que el usuario lo entienda y pueda agruparla.

➢ Seleccionar los participantes

Procurar tener participantes que representen todos los posibles usuarios potenciales. No utilizar la técnica con diseñadores o compañeros que no serán usuarios del sistema. Estar seguros de que los participantes están familiarizados con el vocabulario de las tarjetas. Si tenemos grupos diferentes, realizar sesiones separadas.

Número de participantes

Generalmente diez participantes son suficientes para realizar la prueba y obtener resultados interesantes. Sin embargo, a mayor número de tarjetas y mayor dificultad de categorización, es recomendable un mayor número de participantes.

Aclaraciones a los participantes

Es importante dejar claro a los participantes que pueden hacer tantos montoncitos como deseen y del número de tarjetas que quieran, incluso grupos de una sola tarjeta. Hay que evitar que los usuarios puedan tergiversar los grupos que deban ser similares, al contrario, se debe favorecer que no tengan miedo de mostrar sus propias opiniones.

También se les debe indicar que primero visualicen todas las tarjetas una por una y luego comiencen a hacer los grupos, para que no se precipiten.

Es adecuado recomendarles que no utilicen demasiado tiempo agrupando las tarjetas. Cuando el usuario razona en gran medida las categorías estas no son realistas, pues cuando este navega por el sitio no piensa detenidamente en lo que va a investigar, simplemente busca donde primero estima conveniente.

- Hacer la/s sesión/es de ordenación

Se debe explicar el proceso a seguir. Una explicación por escrito asegura que todos tienen el mismo nivel de comprensión. En el caso del Card Sorting abierto la ordenación es sin grupos preestablecidos, siendo útil para arquitecturas nuevas; sin embargo en el caso del cerrado los grupos de ordenación están predefinidos para arquitecturas ya existentes. También se le remarca a los participantes a mirar todas las etiquetas antes de empezar la ordenación, a organizar las tarjetas utilizando un criterio común y de acuerdo a sus propios principios permitir u grupo de “no estoy seguro” pero no explicarlo al inicio. En una sesión Card Sorting abierto los participantes deben etiquetar los grupos a su manera. El observador observa y escucha, no guiará a los participantes y toma nota de cualquier suceso que pueda serle de importancia como preguntas, comentarios, sugerencias.

Ordenación N º	1
Fecha	20/05/08
Usuario	Yannelys
Criterio	Sabor

Grupos	Dulce: 1,4,8
	Amargo: 3,5
	Salado: 6,7,2

Figura 6 Ejemplo de una tabla del resultado después de la ordenación.

- Almacenar los resultados

Después de haber realizado el ejercicio se almacenan los resultados haciendo un informe tomándolo como premisa para el diseño del sitio. Generalmente los usuarios coinciden en la agrupación de entre el 60% y 80% de las tarjetas. A mayor claridad en el contenido de las tarjetas y menor número de ellas, mayor nivel de coincidencia. Hay contenidos que son inherentemente difíciles de clasificar, pero si tenemos menos de un 60% de coincidencia, entonces hay que revisar las tarjetas. Siempre existe un 20-40% de tarjetas de difícil agrupación, pero es normal; esto es explicado por las diferencias individuales en experiencia y aprendizaje.

¿Qué hacer con los elementos no agrupados?

Ese 20-40% de tarjetas que no han sido agrupadas por suficientes usuarios en el mismo grupo son problemáticas.
Se puede cambiarle el nombre y repetir la prueba. Por otro lado también es permitido agruparlas forzosamente, aunque solo 4 de 10 usuarios las hayan puesto juntas.
Hay que remarcar que siempre existen elementos totalmente inagrupables que cada usuario ha agrupado en una categoría diferente o que han dejado solos en un grupo propio. En ese caso se debe considerar que el elemento no tiene nada que ver con el resto. Forzar la agrupación no tendría sentido. Lo más adecuado es facilitar la localización de este elemento, por ejemplo, situándolo en un nivel superior (incluso a primer nivel), como si fuese una categoría propia.

Categorizar no es la solución, el buscador es siempre mejor

Un margen de error de entre el 20% y 40% que nos da el Card Sorting no es algo aceptable como solución porque significa que ese porcentaje de usuarios o de elementos no serán localizados en nuestro sitio.
La conclusión es que los sistemas de categorización son un elemento de la interfaz que debe existir, pero más como un complemento que como el sistema prioritario de interacción con el sitio.

El buscador debería ser la interfaz básica y más importante del sitio web porque en la mayoría de situaciones los usuarios ya saben lo que buscan y el buscador es la herramienta más rápida y eficiente. Es crítico mejorar el funcionamiento del buscador porque es lo más cercano a un operador humano que nos da respuestas.

Solo en los casos en los que se realiza una búsqueda más abierta o el usuario no sabe exactamente que busca, la categorización es útil y el usuario recurre a los directorios. Sin embargo en la mayoría de los casos el usuario si sabe lo que busca, por eso la interfaz prioritaria es el buscador.

La cantidad y calidad de la información que se pueda extraer del empleo de esta técnica dependerán del tipo de análisis que se realice. Entre los tipos de análisis, se puede distinguir los cualitativos y cuantitativos. El análisis cualitativo se puede realizar cuando los participantes, así como el número de categorías no son muy numerosos.

Consiste, por una parte, en observar de forma individual a cada usuario durante la prueba, y anotar todos los aspectos relativos a cómo cada usuario organiza las tarjetas, qué problemas tiene para realizar la tarea, qué categorías agrupa inmediata e intuitivamente y sobre cuáles duda más, qué preguntas hace durante la prueba, etc.

Después, se analizan detenidamente los grupos creados por los participantes, observando qué categorías guardan más relación con qué categorías según el modelo mental del usuario. El análisis cuantitativo consiste en el procesamiento estadístico de los datos, y el posterior resumen de los resultados a través de representaciones gráficas que faciliten su interpretación por parte del arquitecto de información. Es, por tanto, un análisis adecuado para pruebas con gran número de participantes y categorías a ordenar.

Primero se crea una tabla de co-ocurrencias en una hoja de cálculo. En esta, con tantas filas y columnas como número de categorías diferentes, se indica el número de veces que cada par de categorías han sido colocadas en un mismo grupo, dando como resultado una matriz simétrica como la siguiente.

	Guía de estudio	Des. materiales	Foro	Web personal	Temario	Glosario
Guía de estudio		30	1	1	47	49
Desc. materiales	30		6	9	40	36
Foro	1	6		33	0	1
Web personal	1	9	33		0	0
Temario	47	40	0	0		55
Glosario	49	36	1	0	55	

Tabla Nº2 Análisis estadísticos de las categorías en la web

Después, sobre esta tabla de co-ocurrencias se aplican algoritmos de reducción dimensional, como son los algoritmos de clústering y de escalamiento multidimensional (MDS), cuya función es simplificar las relaciones entre categorías a un número de dimensiones fácilmente interpretables por inspección visual (2D ó 3D).

Como se puede apreciar en las figuras, el resultado de la aplicación del algoritmo de clústering es un dendograma, y en el MDS una representación geométrica de las categorías, distanciadas según la propia similaridad entre estas.

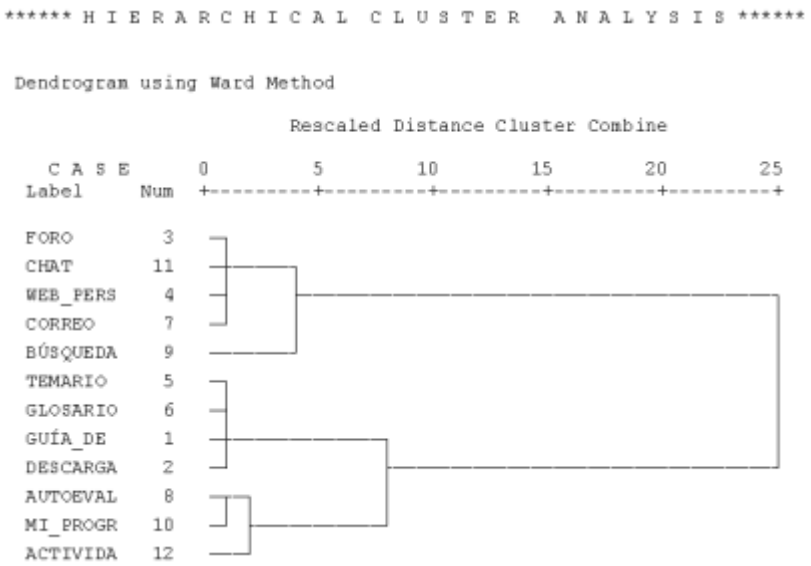


Figura 7- Dendograma (Clústering)

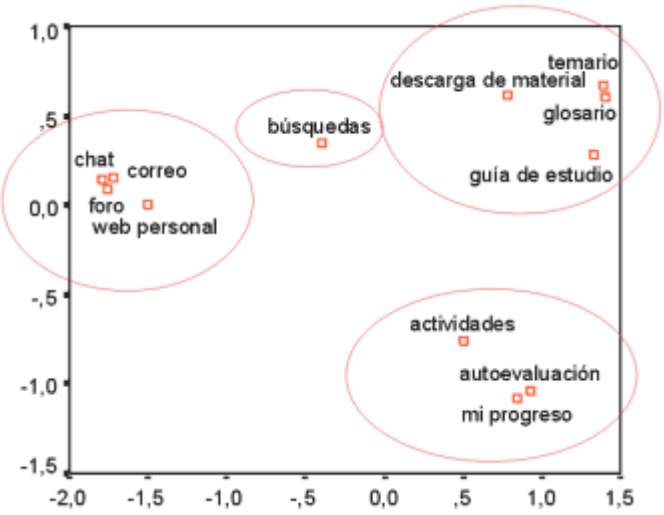


Figura 8- MDS (Escalamiento multidimensional)

2.3 Objeto automatización

Actualmente el proceso de gestión de la información centrada en el usuario a través de la técnica del Card Sorting en la UCI no está automatizado, provocando de esta forma disímiles errores. Para darle solución a esta problemática se decidió automatizar el proceso de crear las tarjetas: es decir definirlas, ponerles nombre y estas luego serán agrupadas; preparar los bloques: es cómo hacer subgrupos; crear clases: consiste en definir las clases que posteriormente se agruparan en bloques; preparar las instrucciones: aquí es donde se elabora una series de preguntas en cuanto a las dudas que la audiencia pueda tener a la hora de realizar el ejercicio; preparar plantillas, aquí es donde la audiencia va poniendo los resultados de los ejercicios ; y almacenar al final las agrupaciones de clases, de bloques y el orden en el caso del ejercicio de secuencia.

2.4 Propuesta del Sistema

Actualmente en la UCI se hace necesario un sistema informático centrado en el usuario que realice la técnica del Card Sorting u Ordenación de Tarjetas, como son muchas actividades a realizar para gestionar los procesos de este mecanismo el sistema debe contar con dos módulos uno para los arquitectos y otro para la audiencia.

El módulo de los arquitectos debe permitir preparar orientaciones y ejercicios de establecimientos de clases, secuencia y bloques, incluyendo la posibilidad de combinarlos, modificarlos y de crear un nuevo proyecto, luego los ejercicios preparados se guardan en el fichero XML, se procesan los resultados y se obtiene un informe que permite ver la solución de los ejercicios resueltos por la Audiencia. El módulo de la audiencia debe permitir la lectura de ficheros XML con los ejercicios preparados en el otro módulo, y realizar los ejercicios indicados, además de poder acceder a ver las orientaciones. Al final debe posibitarle a la audiencia ver la solución y guardar la misma en el fichero XML de ejercicios resueltos.

De esta forma se evitaría la introducción de errores manuales, manteniendo archivados los datos de forma electrónica y garantizando el acceso a los mismos en cualquier momento.

2.5 Modelo de negocio

El modelo de negocio es una técnica que permite comprender los procesos del negocio de la organización. Este proceso es esencial en el desarrollo de cualquier software, con el objetivo de comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema; comprender los problemas actuales de la organización e identificar las mejoras potenciales; al igual que derivar los requerimientos del sistema que va a soportar la organización, logrando una comunicación efectiva entre los usuarios y el equipo de proyecto para llegar así a un entendimiento de lo que hay que hacer, sin dudas se puede decir que es la clave del éxito en la producción de un software.
Para entender el negocio se usó la técnica de la entrevista, pues se tuvo que entrevistar a diferentes Arquitectos de Información para obtener la percepción directa del funcionamiento de la técnica.

2.5.1 Descripción de los procesos del negocio

La técnica del Card Sorting suele realizarse en un local donde un grupo de personas son reunidas en calidad de audiencia, se le suministra un grupo de tarjetas de cartón numeradas y con un concepto en el reverso. La audiencia las agrupa en dependencia del ejercicio que previamente se les orientó realizar. El Card Sorting consta de tres ejercicios:

- Establecimiento de Clases: La audiencia agrupa según el criterio propio las tarjetas en clases.
  - Abierto: La audiencia puede crear tantas clases como crea necesarias.
  - Cerrado: La audiencia no puede crear clases, tiene que ajustarse a las predefinidas por el arquitecto.
- Establecimiento de Secuencia: Se procesa el orden en que la audiencia agrupó las clases.
- Establecimiento de Bloque: La audiencia coloca cada clase en un bloque determinado.

Los resultados de la prueba son recuperados, procesados por el arquitecto de la información y luego se guardan en una carpeta.

2.5.1.1 Actores del negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados. [19]

Actor	Descripción
Audiencia	Se beneficia de realización los ejercicios del Card Sorting.
Diseñador	Se del informe de lo resultados para diseñar la estructura del contenido.

Tabla 3 Actores del Negocio

2.5.1.2 Trabajadores del Negocio

Un trabajador del negocio representa una persona o un sistema automatizado (software) que actúa en el negocio realizando una o varias actividades comprendidas dentro del caso de uso, interactuando con otros trabajadores del negocio y manipulando entidades del negocio. [20]

Trabajador	Descripción
Arquitecto de información	Encargado de gestionar toda la información durante el Card Sorting.
Observador	Encargado de orientar a la audiencia.

Tabla 4 Trabajadores del Negocio

2.5.1.3 Diagrama de Casos de Uso del Negocio

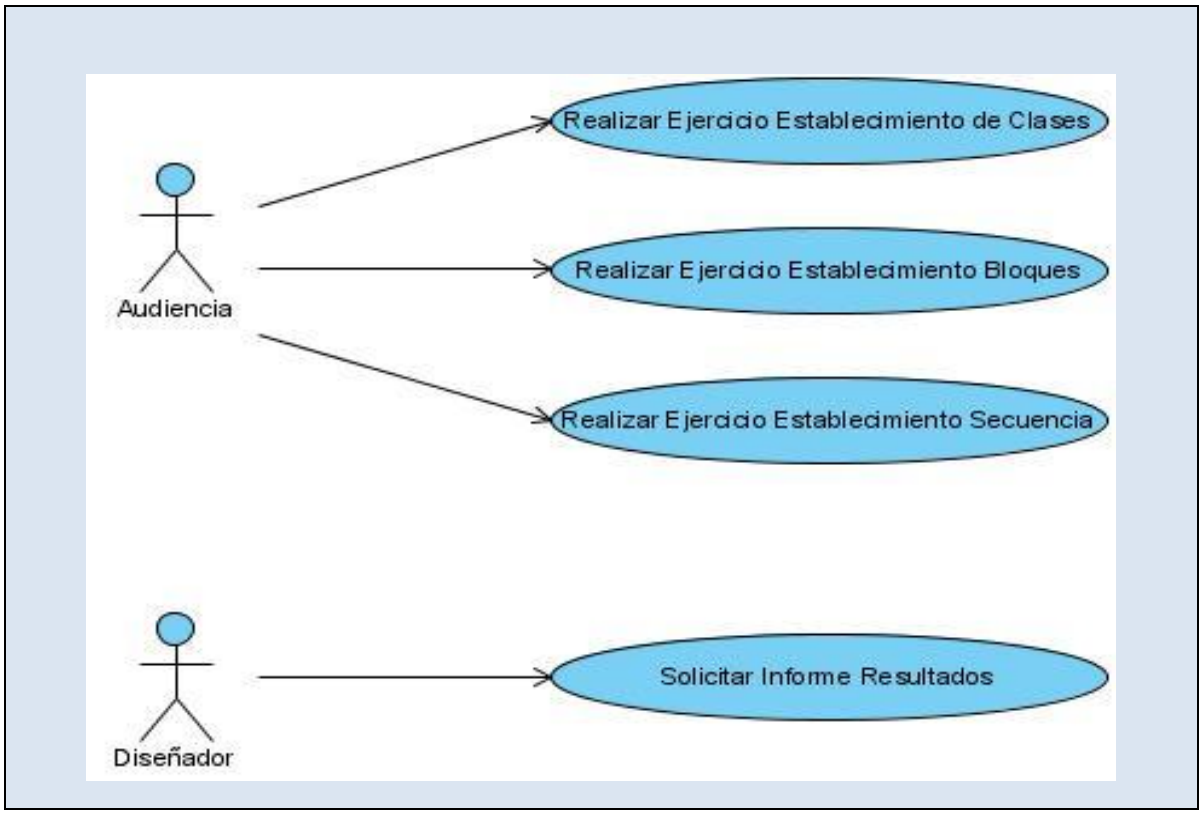
Un caso de uso del negocio representa un proceso dentro del negocio que se estudia, por lo que se corresponde con una secuencia de acciones con un orden lógico y que producen un resultado observable para ciertos actores del negocio.[21]

[19] JACOBSON, I. El proceso Unificado de Desarrollo de Software. La habana, 2004. 438 p.

[20] Op. Cit. (19)

[21] Ibídem

A continuación se muestra un diagrama con la relación entre los actores y casos de uso de negocio que se han identificado durante el proceso de la técnica del Card Sorting.



Casos de Uso del Negocio

Realizar Ejercicio de Establecimiento de Bloques: Dado un ejercicio de bloques la audiencia es instruida para garantizar el éxito del ejercicio y seguidamente organiza el contenido de las clases en la pantalla, obteniéndose como resultados los bloques.

Realizar Ejercicio de Establecimiento de Secuencia: Si la audiencia escoge la posibilidad de realizar el ejercicio por secuencia, antes es instruido para estar orientado en lo que va a realizar y luego agrupa las clases basándose en su criterio, en este caso es importante el orden dado en ese agrupamiento para después recuperar la información en el mismo orden que se hizo el ejercicio.

Realizar Ejercicio de Establecimiento de Clases: Si la opción a realizar el ejercicio por la audiencia es de clases, se le explica en que consiste este mecanismo, luego en caso de ser un Card Sorting cerrado la audiencia solo agrupará las tarjetas en las clases predefinidas, en caso de ser un Card Sorting abierto la audiencia agrupará las tarjetas en las clases predefinidas o en las creadas por el mismo.

Solicitar Informe Resultados: En este caso el diseñador solicita al arquitecto de información un informe sobre los resultados del ejercicio después de haber sido procesados los mismos.

2.6 Descripción de los Casos de Uso del Negocio

Realizar Ejercicio Establecimiento de Bloques

Caso de Uso:	Realizar Ejercicio Establecimiento de Bloques
Actores:	Audiencia
Trabajadores:	Observador y Arquitecto de Información
Resumen:	El caso de uso comienza cuando la audiencia realiza el agrupamiento por bloques de las clases, luego son guardados en una carpeta, terminando de esta manera el caso de uso.
Precondiciones:	La audiencia debe ser instruida para realizar el ejercicio y además debe poseer las clases y los bloques para agrupar.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. La Audiencia agrupa las clases en una tarjeta de cartón.	2. El Observador analiza detalladamente los grupos de clases.
	3. El Arquitecto guarda los grupos de clases en la hoja de plantilla de resultados.
4. La Audiencia agrupa las clases en bloque que es una tarjeta de cartón.	5. El observador revisa los grupos de clases que están dentro de los bloque y guarda los resultados en una carpeta, finalizando así el Caso de Uso.
Poscondiciones	Quedan establecidos los bloques, que deben ser recuperados.

Realizar ejercicio Establecimiento de Secuencia

Caso de Uso:	Realizar ejercicio Establecimiento de Secuencia
Actores:	Audiencia

Trabajadores:	Observador y Arquitecto de Información
Resumen:	El caso de uso comienza cuando la audiencia agrupa las tarjetas de clases basándose en su criterio propio, en este caso es importante el orden dado en ese agrupamiento por lo que debe ser recuperado, finalizando de esta forma el caso de uso.
Precondiciones:	La audiencia debe ser instruida para realizar el ejercicio y debe poseer las clases

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. La Audiencia ordena las tarjetas.	2. El Observador analiza detalladamente el orden de las tarjetas.
3. La Audiencia agrupa las tarjetas en clases basándose en su criterio.	4. El Observador revisa los grupos de tarjetas en las clases.
	5 El Arquitecto guarda en la hoja de plantilla de los resultados el agrupamiento de las tarjetas en clases y posteriormente es guardado en una carpeta.
Poscondiciones	Quedan establecidas las secuencias que deben ser recuperadas

Realizar ejercicio Establecimiento de Clases

Caso de Uso:	Realizar ejercicio Establecimiento de Clases
Actores:	Audiencia
Trabajadores:	Observador y Arquitecto de Información
Resumen:	El caso de uso comienza cuando la Audiencia va a realizar un ejercicio, en caso de ser un Card Sorting abierto la audiencia agrupará las tarjetas en las clases predefinidas o en clases creadas basándose en su criterio. En caso de ser un Card Sorting cerrado la audiencia solo agrupará las tarjetas en clases predefinidas, finalizando así el caso de uso.
Precondiciones:	La audiencia debe ser instruida para realizar el ejercicio y debe poseer

	las tarjetas y clases.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. La Audiencia crea tantas clases nuevas como estime conveniente basándose en su criterio.	
2. La Audiencia agrupa las tarjetas en las clases ya sean las predeterminadas o las creadas por ella misma.	3. El Observador analiza detalladamente el agrupamiento de las tarjetas en clases.
	4. El Arquitecto almacena los agrupamientos (clases) de tarjetas, en la hoja de plantilla de recuperación de resultados y posteriormente es guardado en una carpeta.
Flujos Alternos	
Acción del Actor	**Respuesta del Negocio**
1. La audiencia agrupa las tarjetas en las clases predefinidas basándose en su criterio.	2. El Observador analiza detalladamente el agrupamiento de las tarjetas en las clases predefinidas.
	3. EL Arquitecto almacena los agrupamientos (clases) de tarjetas, en la hoja de plantilla de recuperación de resultados, y posteriormente lo guarda en una carpeta.
Poscondiciones	Quedan establecidas las clases para su recuperación

Solicitar Resultados

Caso de Uso:	Solicitar Resultados
Actores:	Diseñador
Trabajadores:	Arquitecto de la información y Observador
Resumen:	El caso de uso comienza cuando el Diseñador solicita al Arquitecto de información un informe sobre los resultados de la aplicación del Card

	Sorting. El Arquitecto de información procesa los resultados recuperados sumando la cantidad de veces que se repitió una misma categoría, terminando con esto el Caso de Uso.
Precondiciones:	Se deben haber realizado el ejercicio(s) de Card Sorting previamente.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. El diseñador solicita informe de los resultados de los ejercicios.	2. El Observador tabula los resultados de los ejercicios, sumando la cantidad de veces que se repitió una misma categoría.
	3. El Arquitecto de Información obtiene los patrones más persistentes entre la audiencia, es decir las categorías que mas se repitieron en la realización del ejercicio.
4. El diseñador solicita un informe de los resultados	5. El Arquitecto redacta el informe en una hoja y lo guarda en una carpeta.
Poscondiciones	Se obtienen los patrones más persistentes entre la audiencia.

2.7 Diagramas de Actividades

Diagrama de Actividades: Realizar Ejercicio Establecimiento de Clases

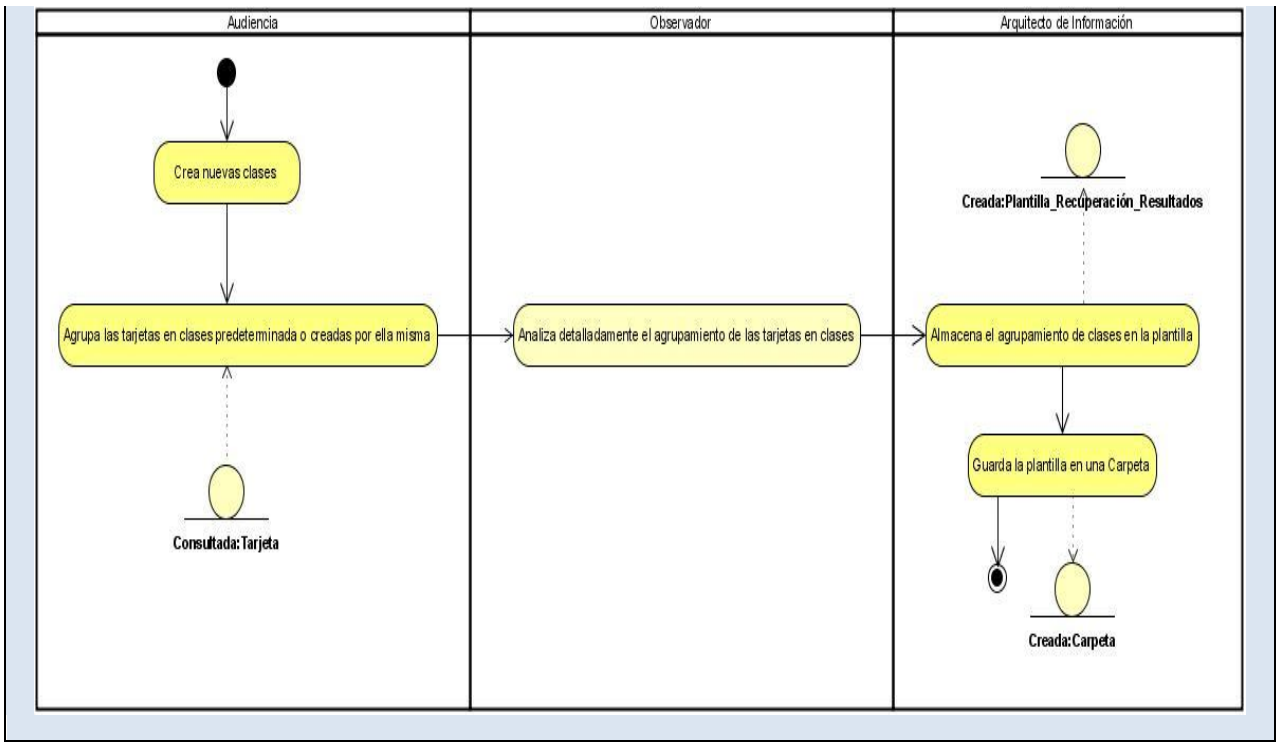

Audiencia
Observador
Arquitecto de Información
Crea nuevas clases
Agrupa las tarjetas en clases predeterminada o creadas por ella misma
Consultada:Tarjeta
Analiza detalladamente el agrupamiento de las tarjetas en clases
Creada:Plantilla_Recuperación_Resultados
Almacena el agrupamiento de clases en la plantilla
Guarda la plantilla en una Carpeta
Creada:Carpeta

Diagrama de Actividades: Realizar Ejercicios de Establecimiento de Bloques

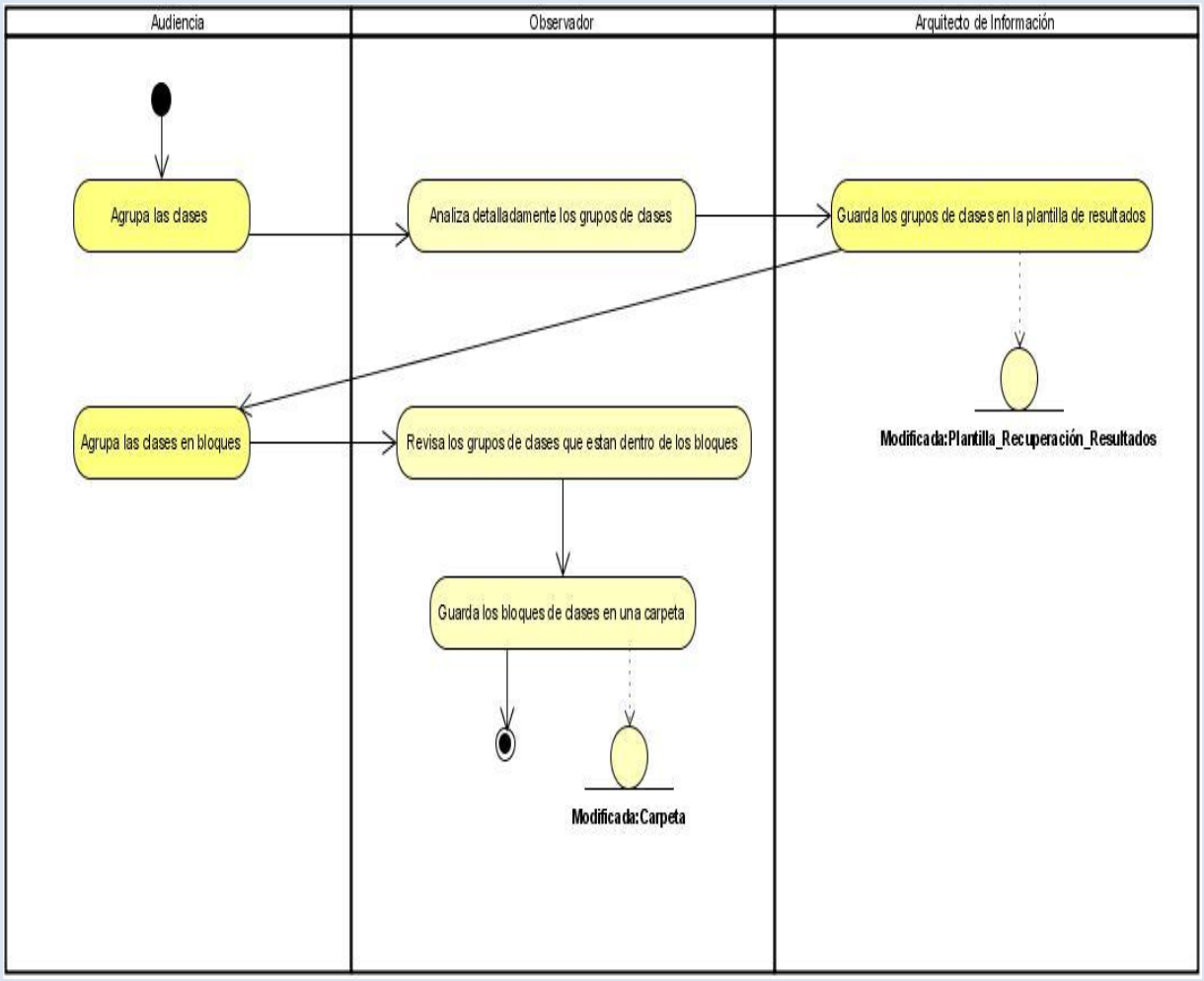


Diagrama de Actividades: Realizar Ejercicio de Establecimiento de Secuencia

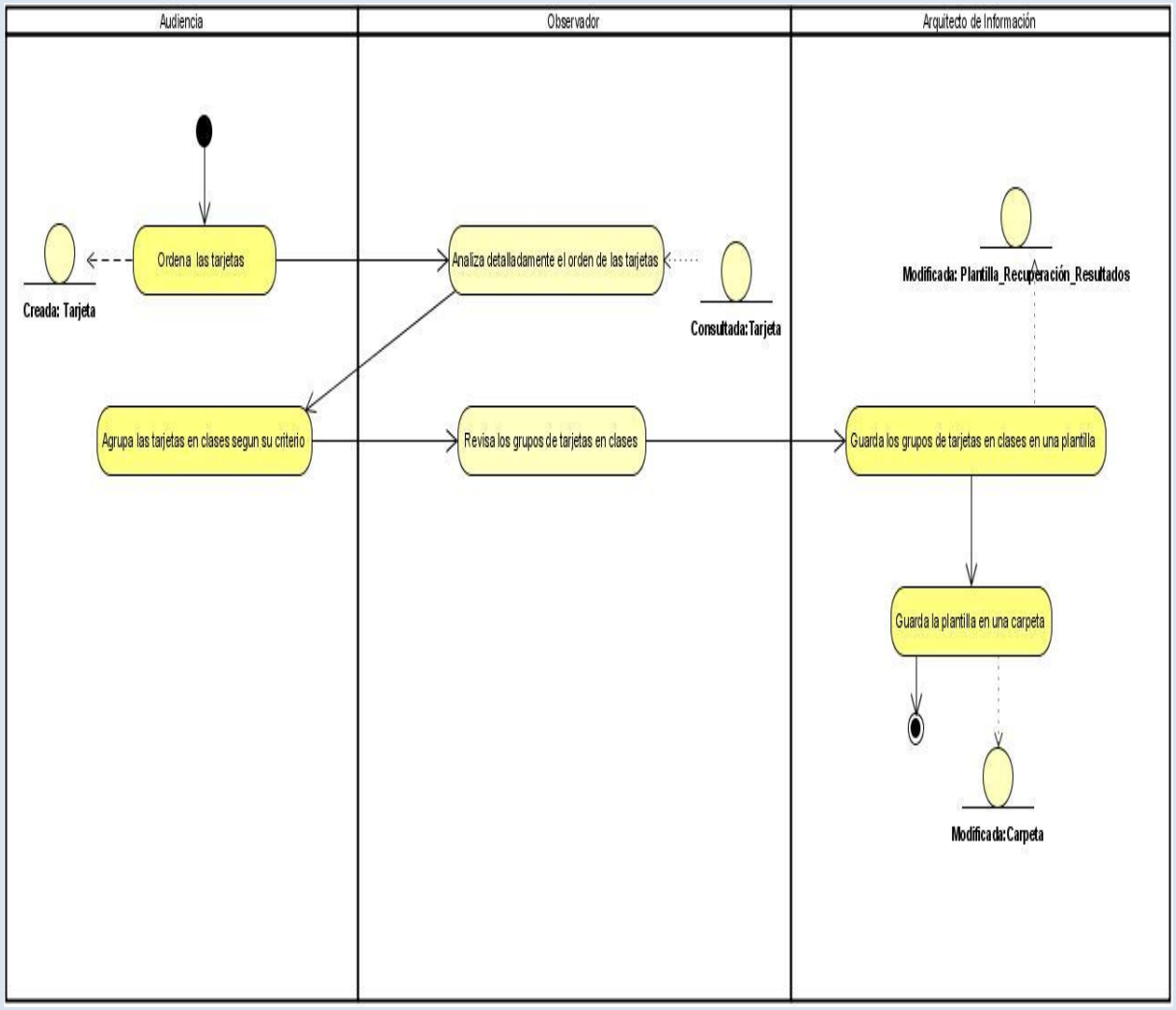
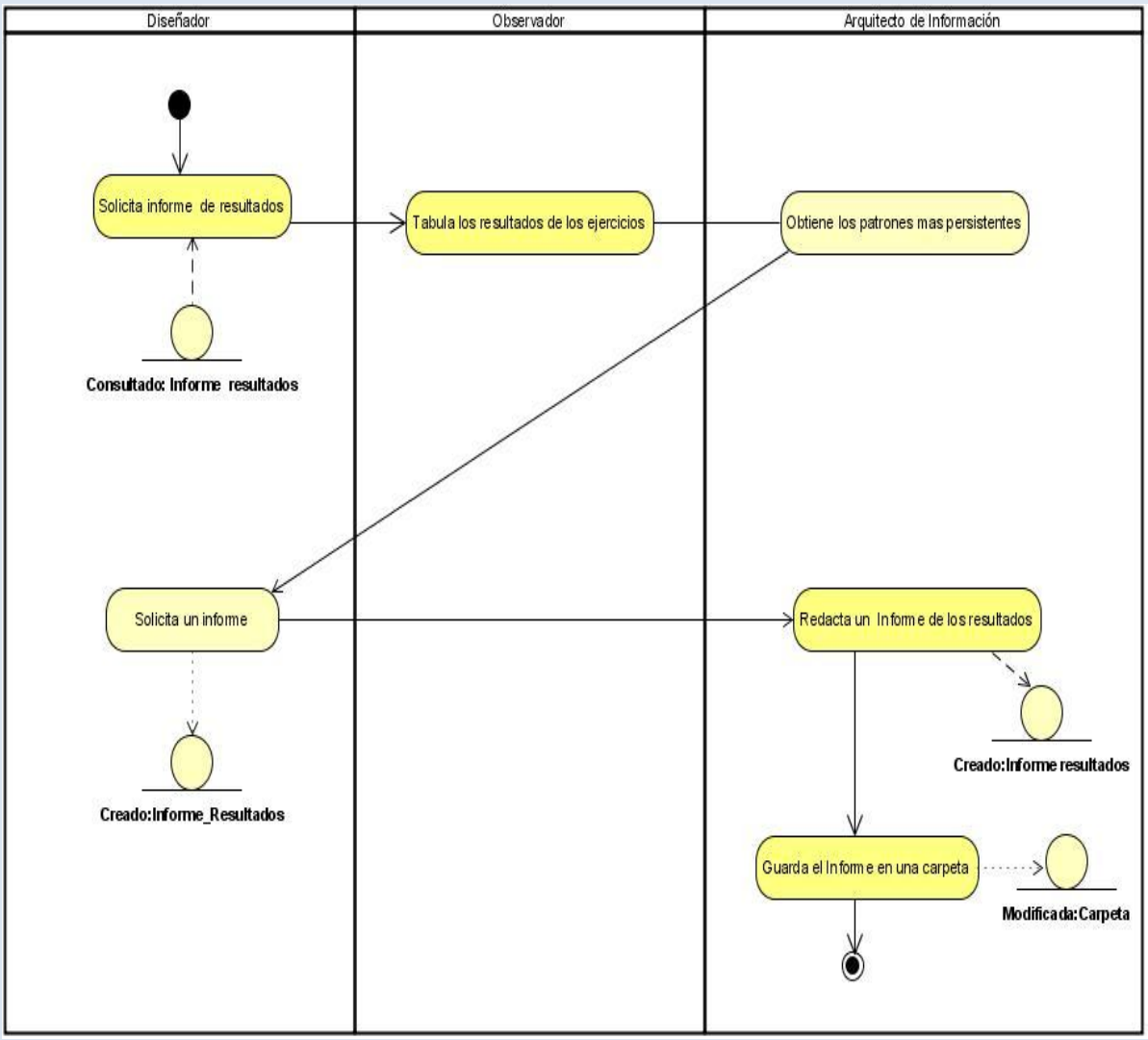
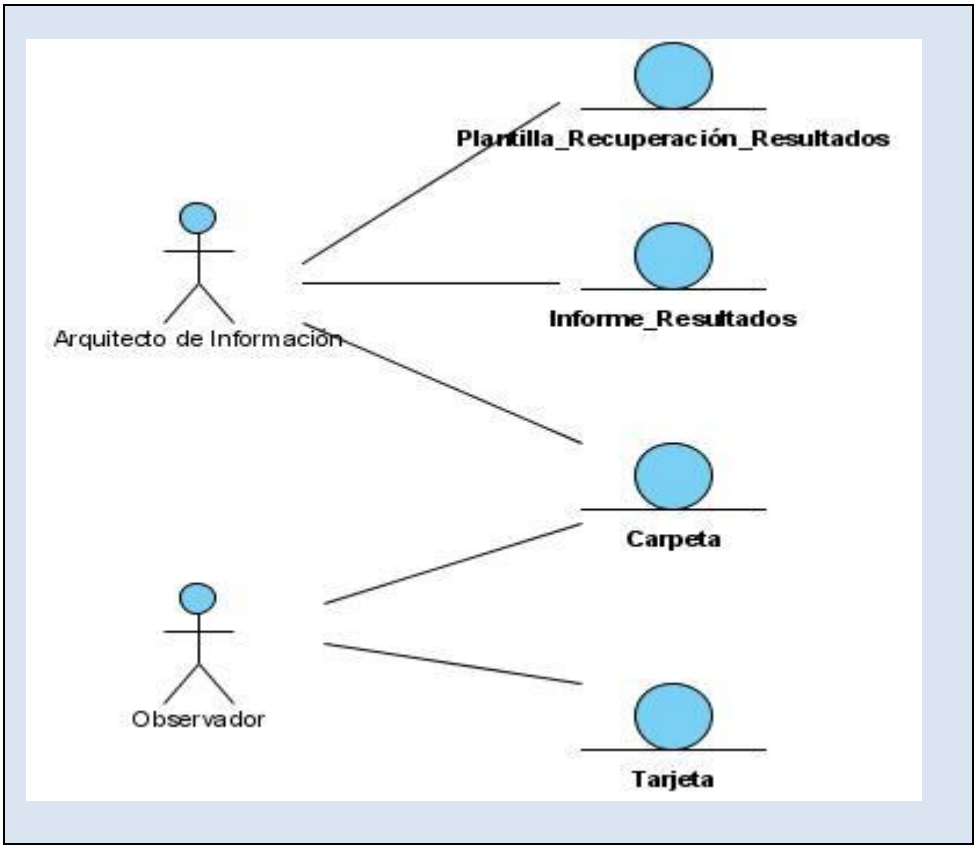


Diagrama de Actividades: Solicitar Resultados



2.8 Diagrama de Clase Modelo de Objeto



2.9 Levantamiento de requisitos

Se desarrolla en la fase de Inicio del RUP y tiene como entrada el modelamiento de negocio. Su resultado final una vez concluida la fase es tener acabado el Modelo del sistema.

El levantamiento de los requisitos tiene como objetivos:

- Identificar, enunciar y clasificar los requerimientos.
- Obtener las propiedades o cualidades que el producto debe cumplir.
- Obtener un listado de las capacidades o funciones que el sistema debe cumplir.
- Determinar actores y casos de uso del sistema.

2.9.1 Requisitos Funcionales

Los requisitos funcionales son condiciones o capacidades que el sistema debe cumplir. Este sistema debe ser capaz de:

RF 1. Preparar Orientaciones de los ejercicios de Card Sorting.
RF 2. Preparar Clases de la temática de la web a diseñar.
RF 3. Preparar Tarjetas con el contenido del sitio.
RF 4. Permitir crear Tarjetas con el contenido necesario que se estime conveniente.
RF 5. Permitir que se creen Clases.
RF 6. Permitir preparar Ejercicios de Establecimiento de Clases.
RF 7. Permitir preparar Ejercicios de Establecimiento de Bloques.
RF 8. Permitir preparar Ejercicios de Establecimiento de Secuencia.
RF 9. Mostrar Ejercicios preparados.
RF10. Permitir guardar Ejercicios Preparados en el fichero XML de Ejercicios Preparados.
RF11. Mostrar orientaciones de los Ejercicios Preparados.
RF12. Permitir realizar Ejercicios de Establecimiento de Clases Abierto.
RF13. Permitir actualizar Ejercicios de Establecimiento de Clases Abierto.
RF14. Permitir realizar Ejercicios de Establecimiento de Clases Cerrado.
RF15. Permitir actualizar Ejercicios de Establecimiento de Clases Cerrado.
RF16. Permitir realizar Ejercicios de Establecimiento de Clases Mixto.
RF17. Permitir actualizar Ejercicios de Establecimiento de Clases Mixto.
RF18.Permitir realizar Ejercicios de Establecimiento de Bloques.
RF19. Permitir actualizar Ejercicios de Establecimiento de Bloques.
RF20. Permitir realizar Ejercicios de Establecimiento de Secuencia.
RF21. Permitir actualizar Ejercicios de Establecimiento de Secuencia.
RF22. Permitir ver la solución de los ejercicios.
RF23.Permitir guardar Ejercicios Realizados en el Fichero XML de Ejercicios Resueltos.
RF24.Permitir modificar un proyecto de Card Sorting (nombre, identificador, categoría, tipo de ejercicio (Establecimiento de Clases, Bloques y Secuencia))
RF25. Permitir crear un nuevo proyecto de Card Sorting (nombre, identificador, categoría, tipo de ejercicio (Establecimiento de Clases, Bloques y Secuencia))
RF26. Permitir procesar los resultados de los ejercicios aplicados a la Audiencia
RF27. Permitir aplicar el algoritmo de clústering a los ejercicios.
RF28 Mostrar resultados finales de los ejercicios.
RF29 Permitir obtener el Informe de los resultados finales.

RF30 Mostrar Plantilla de resultados de los ejercicios aplicados a la Audiencia.
RF31 Permitir guardar los resultados de los ejercicios en el fichero XML de Resultados Finales.

2.9.2 Requisitos No Funcionales

Los requisitos no funcionales son propiedades o cualidades que el sistema debe tener para su máximo rendimiento.

- Software

El sistema debe disponer de un Sistema Operativo Windows 98 o superior y esta implementado en Visual Studio 2003.

- Apariencia o Interfaz Interna

El software debe contar con una interfaz fácil, amigable, sencilla, permitiendo que la audiencia sea capaz de interactuar con este aún teniendo conocimientos básicos y se sienta familiarizado con el sistema.

- Usabilidad

Se refiere a la facilidad de uso por parte del usuario; a la capacidad del sistema de ser comprendido, aprendido y usado por este usuario. En este caso, el sistema será flexible y de fácil aprendizaje, pues se trata en todo lo posible de mantener un estándar de operatividad que logre que las interacciones del usuario con el sistema sean predecibles y familiares. El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora.

- Rendimiento

Las pantallas estarán poco cargadas de imágenes para garantizar la ejecución de los hipervínculos, las adiciones, modificaciones no excedan los 4 segundos y garantizar de esta manera una respuesta rápida del sistema.

- Soporte

Debe elaborarse un paquete de instalación que abarque verificación de componentes ya instalados y la instalación de los nuevos.

- Portabilidad

La portabilidad se refiere a la capacidad que tienen los programas de ejecutarse en diferentes sistemas operativos con mínimas modificaciones. El sistema podrá ejecutarse sobre plataforma Linux, Windows 98 o superior.

- Seguridad

La herramienta garantiza que la información sea actualizada únicamente por quien tiene acceso a trabajar con el sistema. Tiene protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

- Confiabilidad

La herramienta a utilizar tiene un mecanismo eficiente para recuperar las pérdidas de información ante fallos, errores en caso de alguna ocurrencia, por lo que deberá existir un plan de salvas y mantenimiento garantizando con esto una rápida protección y recuperación ante un problema dado. También debe montarse sistemas de respaldo eléctrico en los locales de los servidores para mantener la vitalidad de los servicios.

- Interfaz

El diseño de la aplicación debe ser con colores claros, adecuado y lo mas legible posible.

- Ayuda y documentación en línea

La aplicación debe contar con una ayuda, así la audiencia tendrá conocimiento de las funcionales del mismo y hacer un mejor uso de este.

2.10 Definición de los casos de usos

2.10.1 Actores del sistema

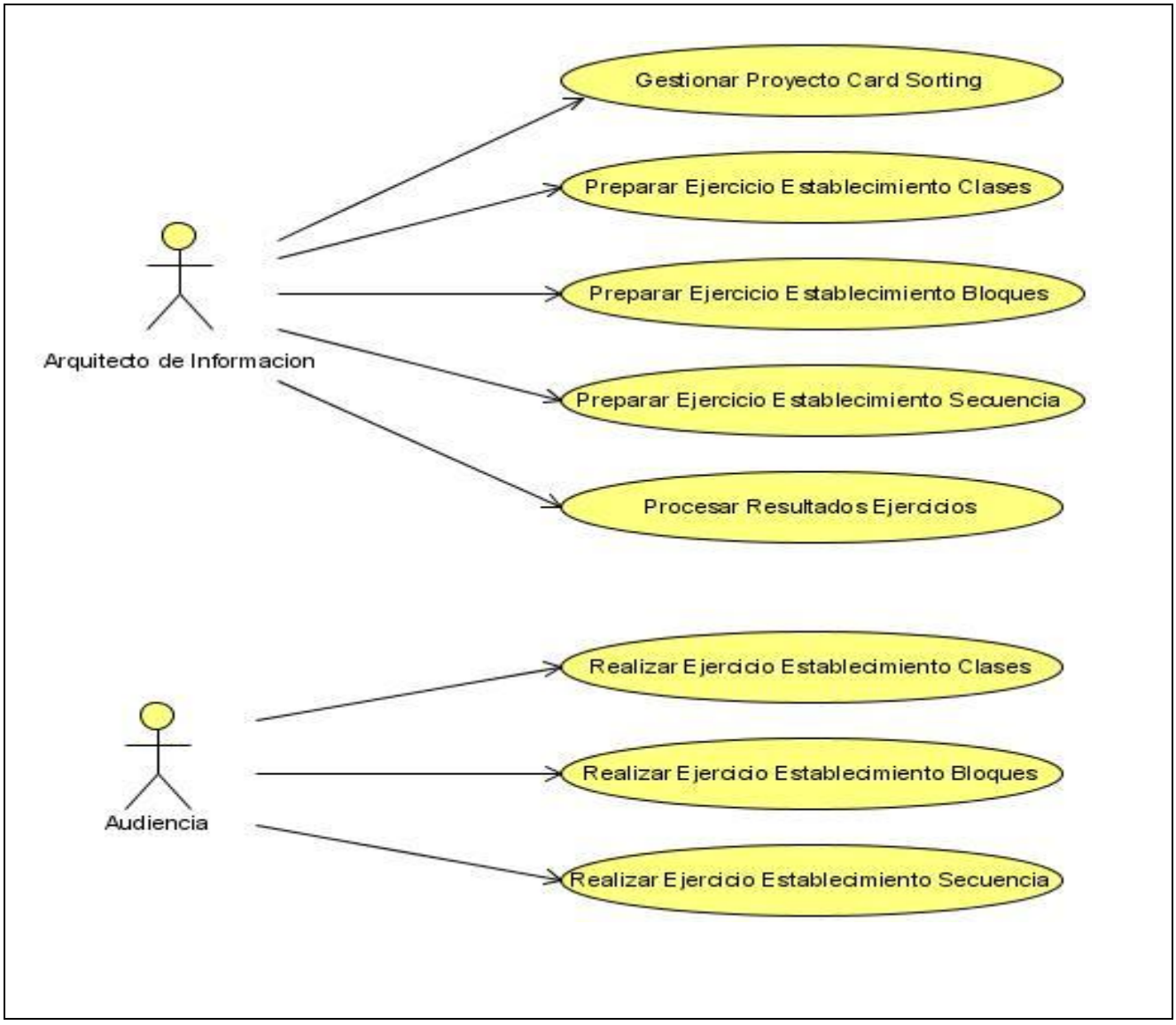
Los actores del sistema representan entidades externas que interactúan directamente con el sistema (personas, máquinas u otros sistemas). [22]

Actor	Descripción
Arquitecto de Información	Es el encargado de preparar todos los tipos de ejercicios, de recoger los resultados de los mismos resueltos con anterioridad por la audiencia.
Audiencia	Es el encargado de realizar cada uno de los ejercicios que el arquitecto hace.

[22] Op. Cit(19)

Tabla 3 Actores del Sistema

2.11 Diagrama de Casos de Uso del Sistema



2.12 Casos de Uso Expandidos

Gestionar Proyecto de Card Sorting

Caso de Uso:	Gestionar Proyecto de Card Sorting
Actores:	Arquitecto de Información
Resumen:	El Caso de Uso comienza cuando el Arquitecto de Información prepara un proyecto de Card Sorting, este puede modificar uno existente o crearlo nuevo, al igual que confecciona las orientaciones para los ejercicios y los guarda en un fichero XML de Ejercicios Preparados, finalizando así el caso de uso.
Precondiciones:	
Referencias	RF 10, RF 24, RF 25,
Prioridad	Crítico

Flujo Normal de Eventos	
Sección "Modificar Proyecto"	
Acción del Actor	**Respuesta del Sistema**
1. El Arquitecto accede a modificar un determinado dato del Proyecto de Card Sorting.	2. El sistema muestra un formulario con varios criterios de búsqueda como el nombre, identificador y categoría del proyecto.
3. El Arquitecto introduce datos en los criterios de búsqueda.	4. El sistema solicita el proyecto que cumpla con el criterio de búsqueda seleccionado.
	5. El sistema muestra el formulario con los datos del proyecto encontrado (nombre, identificador, categoría, tipo de ejercicio (Establecimiento de Clases, Bloques y Secuencia)).
6. El Arquitecto selecciona modificar el Proyecto de Card Sorting encontrado.	7. El sistema muestra todos los campos del proyecto buscado permitiendo que sean modificados.
8. El Arquitecto modifica los campos necesarios y selecciona actualizar.	9. El sistema verifica que todos los campos obligatorios estén llenos.
	10. El sistema le solicita que actualice el Proyecto en el fichero XML de Ejercicios Preparados.
	11. El sistema muestra un mensaje informando que el Proyecto ha sido modificado satisfactoriamente

	finalizando el Caso de Uso.
Flujos Alternos	
Acción 4	Se emite un mensaje informando que el proyecto buscado no se encuentra registrado.
Acción 9	Se emite un mensaje para que se llenen todos los campos obligatorios.
Sección "Crear Proyecto"	
Acción del Actor	**Respuesta del Sistema**
1. El arquitecto accede a crear un nuevo proyecto de Card Sorting.	2. El sistema muestra un formulario con todos los datos del proyecto (identificador, nombre, categoría, tipo de ejercicio (Establecimiento de Clases, Bloques y Secuencia)).
3. El arquitecto inserta los datos (nombre, categoría, tipo de ejercicio (Establecimiento de Clases, Bloques y Secuencia)) del nuevo proyecto y selecciona actualizar.	4. El sistema verifica que todos los campos obligatorios estén llenos.
	5. El sistema solicita que se actualice el Proyecto en el fichero XML de Ejercicios Preparados.
	6. El sistema muestra un mensaje informando que se ha guardado el nuevo proyecto en el fichero XML de Ejercicios Preparados, finalizando así el Caso de Uso.
Flujos Alternos	
Acción 4	Se emite un mensaje para que se llenen todos los campos obligatorios.
Poscondiciones	Queda guardado el Proyecto de Card Sorting en su conjunto.

Preparar Ejercicio de Establecimiento de Clases

Caso de Uso:	Preparar Ejercicio de Establecimiento de Clases

Actores:	Arquitecto de Información
Resumen:	El Caso de Uso comienza cuando el Arquitecto de Información prepare el Ejercicio de Establecimiento de Clases luego lo almacena en el fichero XML de Ejercicios Preparados, finalizando así el Caso de Uso.
Precondiciones:	Debe existir un proyecto creado en el fichero.
Referencia	RF 1,RF 2,RF 3,RF 6,RF 10
Prioridad	Critico

Flujo Normal de Eventos	
Sección "Inicial"	
Acción del Actor	**Respuesta del Sistema**
1. El Arquitecto accede a preparar las orientaciones y el ejercicio de establecimiento de clases para la Audiencia.	2. El sistema muestra un formulario para redactar las orientaciones
3. El Arquitecto redacta las orientaciones y selecciona guardar.	4. El sistema lo guarda y muestra una interfaz con las orientaciones redactadas dando la opción de crear una nueva clase.
5. El Arquitecto selecciona crear una nueva clase.	6. El sistema muestra un formulario con los datos (nombre, categoría) de las clases a crear.
7. El Arquitecto inserta los datos (nombre, categoría) de las clases.	8. El sistema verifica que todos los campos obligatorios estén llenos.
9. El Arquitecto selecciona guardar y actualiza las clases creadas.	10. El sistema le solicita que actualice las clases del Proyecto en el fichero XML de Ejercicios Preparados.
	11. El sistema da la posibilidad de crear la tarjeta de dicha clase.
12. El Arquitecto accede a crear las tarjetas necesarias para el ejercicio.	13. El sistema le muestra un formulario con los datos de las tarjetas a crear (nombre, término, concepto asociado).

14. El Arquitecto inserta los datos (nombre, término, concepto asociado) que le pondrá a las tarjetas.	15. El sistema verifica que todos los campos estén llenos.
16. El Arquitecto selecciona guardar y actualizar las tarjetas.	17. El sistema le solicita que actualice las tarjetas del Proyecto en el fichero XML de Ejercicios Preparados.
	18. El sistema muestra un mensaje informando que se ha guardado el ejercicio de establecimiento de clases en el fichero XML de Ejercicios Preparados, finalizando así el Caso de Uso.
Flujos Alternos	
Acción 8 y 14	Se emite un mensaje para que se llenen los campos obligatorios.
Poscondiciones	Queda preparado el Ejercicio de Establecimiento de Clases.

Preparar Ejercicio de Establecimiento de Secuencia

Caso de Uso:	Preparar Ejercicio de Establecimiento de Secuencia
Actores:	Arquitecto de Información
Resumen:	El Caso de Uso comienza cuando el Arquitecto de Información prepare el ejercicio de Establecimiento de Secuencia, los guarda en el fichero XML de Ejercicios Preparados, finalizando así el Caso de Uso.
Precondiciones:	Deben estar creadas las clases a la que se le aplicará el Establecimiento de Secuencia.
Referencias	RF 1, RF 8, RF 10
Prioridad	Critico
Flujo Normal de Eventos	
Sección "Inicial"	
Acción del Actor	**Respuesta del Sistema**
1. El Arquitecto accede a preparar las orientaciones y el establecimiento de	2. El sistema muestra un Formulario para redactar las orientaciones del ES (texto de

secuencia (ES).	orientaciones)
3. El Arquitecto redacta las orientaciones y selecciona guardar.	4. El sistema lo guarda y muestra una interfaz con las orientaciones redactadas.
	5. El sistema accede al fichero XML de ejercicios preparados para obtener los datos necesarios (clases, tarjetas).
6. El Arquitecto introduce los datos necesarios (clases, tarjetas) y selecciona mostrar una secuencia de ordenación.	7. El sistema muestra una interfaz con la secuencia de ordenación.
8. El Arquitecto selecciona guardar dicha secuencia de ordenación.	9. El sistema lo guarda en el fichero XML de ejercicios preparados y muestra una interfaz con la secuencia, finalizando así el Caso de Uso.
Poscondiciones	Queda preparado el Ejercicio de Establecimiento de Secuencia sobre los ejercicios seleccionados.

Preparar Ejercicio de Establecimiento de Bloques

Caso de Uso:	Preparar Ejercicio de Establecimiento de Bloques
Actores:	Arquitecto de Información
Resumen:	El Caso de Uso comienza cuando el Arquitecto de Información prepara ejercicios de Establecimiento de Bloques. El sistema le permite seleccionar las clases ya confeccionadas destinadas para el mismo, almacenándolos en el fichero XML de ejercicios preparados, finalizando de esta manera el Caso de Uso.
Precondiciones:	Las clases deben de estar creadas para realizar el Establecimiento de Bloques.
Referencias	RF 1,RF 7, RF 10
Prioridad	Crítico

Flujo Normal de Eventos	
Sección "Inicial"	
Acción del Actor	**Respuesta del Sistema**
1. El Arquitecto accede a preparar las orientaciones y el ejercicio de establecimiento de bloques.	2. El sistema muestra un Formulario para redactar las orientaciones de ejercicio de establecimiento de bloque.
3. El Arquitecto redacta las orientaciones y selecciona guardar.	4. El sistema lo guarda y muestra una interfaz con las orientaciones redactas.
	5. El sistema muestra en la misma interfaz un formulario que permite confeccionar un diseño de un bloque dando la opción de seleccionar las clases a utilizar.
6. El Arquitecto diseña un bloque.	
7. El Arquitecto solicita las clases que va utilizar.	8. El sistema muestra una interfaz con todas las clases creadas en el fichero XML de ejercicios preparados.
9. El Arquitecto selecciona las clases que van a participar en el ejercicio.	10. El sistema muestra una interfaz con las clases seleccionadas dando la posibilidad de agruparlas en los bloques diseñados.
11. El Arquitecto agrupa las clases en bloques y selecciona guardar.	12. El sistema guarda y muestra una interfaz con los bloques creados, finalizando así el Caso de Uso.
Poscondiciones	Queda guardado el Ejercicio de Establecimiento de Bloques.

Resolver Ejercicio de Establecimiento de Bloques

Caso de Uso:	Resolver Ejercicio de Establecimiento de Bloques
Actores:	Audiencia
Resumen:	El Caso de Uso comienza cuando la Audiencia resuelve el Ejercicio de Establecimiento de Bloques, y finaliza guardando la solución en el fichero de

	XML de Ejercicios Realizados.
Precondiciones:	Se debe disponer de los Ejercicios de Card Sorting y orientaciones ya preparados por el Arquitecto de Información.
Referencias	RF 9, RF 11, RF 18, RF 19, RF 22, RF 23
Prioridad	Secundario

Flujo Normal de Eventos	
Sección "Inicial"	
Acción del Actor	**Respuesta del Sistema**
1. La Audiencia accede a realizar el Ejercicio.	2. El sistema muestra una interfaz con los ejercicios y las orientaciones correspondientes al Ejercicio de Card Sorting.
3.La Audiencia selecciona un Ejercicio	4. El sistema muestra en la misma interfaz el ejercicio seleccionado.
	5. El sistema le permite organizar el contenido de las clases en Bloques en la pantalla.
6. La Audiencia organiza el contenido de las clases en Bloques situándolo en la pantalla y selecciona actualizar.	7. El sistema muestra un mensaje informando el estado del ejercicio hasta el momento
8. La Audiencia informa que ha culminado con el ejercicio y solicita ver la solución.	9. EL sistema muestra una interfaz con la solución del ejercicio.
10. La Audiencia revisa la solución y selecciona guardar.	11. El sistema guarda la solución en un fichero XML de Ejercicios Realizados, finalizando así el Caso de Uso.
Poscondiciones	Queda resuelto el Ejercicio de Establecimiento de Bloques.

Resolver Ejercicio de Establecimiento de Clases

Caso de Uso:	Resolver Ejercicio de Establecimiento de Clases
Actores:	Audiencia

Resumen:	El Caso de Uso comienza cuando la Audiencia resuelve el Ejercicio de Establecimiento de Clases, este puede hacerlo por Establecimiento de Clases Abierto, Cerrado o Mixto, luego guarda el ejercicio en el fichero XML de Ejercicios Realizados, terminando así el Caso de Uso.
Precondiciones:	Se debe disponer de los Ejercicios de Card Sorting y orientaciones ya preparados por el Arquitecto de Información.
Referencias	RF 4, RF 5, RF 11, RF 12, RF 13, RF 14, RF 15, RF 16, RF 17, RF22 , RF23
Prioridad	Crítico

Flujo Normal de Eventos	
Sección "Inicial"	
Acción del Actor	**Respuesta del Sistema**
1. La Audiencia decide realizar un Ejercicio de Card Sorting (Abierto, Cerrado o Mixto).	El sistema ejecuta las siguientes acciones: 1.1 En caso de seleccionar Resolver un Ejercicio de Establecimiento de Clases Abierto ir a la sección "Resolver Ejercicio de Establecimiento de Clases Abierto". 1.2 En caso de seleccionar Resolver un Ejercicio de Establecimiento de Clases Cerrado ir a la sección "Resolver Ejercicio de Establecimiento de Clases Cerrado". 1.3 En caso de seleccionar Resolver un Ejercicio de Establecimiento de Clases Mixto ir a la sección "Resolver Ejercicio de Establecimiento de Clases Mixto".
Sección "Resolver Ejercicio de Establecimiento de Clases Abierto"	
Acción del Actor	**Respuesta del Sistema**
1. La Audiencia accede a realizar el Ejercicio.	2. El sistema muestra una interfaz con los ejercicios, las orientaciones, las clases y tarjetas creadas.
	3. El sistema brinda la posibilidad de crear más clases y tarjetas con conceptos de la temática del sitio a diseñar.

4. La Audiencia lee las orientaciones y selecciona un ejercicio.	5. El sistema muestra una interfaz con el ejercicio seleccionado.
	6. El sistema le permite comenzar a colocar las tarjetas en Clases.
7. La Audiencia accede a realizar el ejercicio y agrupa las tarjetas en clases.	8. El sistema muestra un mensaje informando el estado del ejercicio hasta el momento.
	9. El sistema le permite crear nuevas clases.
10. La Audiencia solicita crear una nueva Clase, la nombra y selecciona actualizar y guardar.	11. El sistema muestra una interfaz con el ejercicio actualizado.
12. La Audiencia agrupa libremente las Clases en las categorías que crea necesario y selecciona actualizar.	13. El sistema muestra un mensaje con el estado del ejercicio.
14. La Audiencia informa que ha terminado el ejercicio.	15. El sistema recepciona los datos del ejercicio.
16. La Audiencia solicita ver la solución del ejercicio.	17. El sistema muestra una interfaz con la información del ejercicio resuelto.
18. La Audiencia revisa la solución y selecciona guardar.	19. El sistema informa que se guardo el ejercicio en el fichero XML de Ejercicios Realizados, finalizando así el Caso de Uso.
Sección " Resolver Ejercicio de Establecimiento de Clases Cerrado"	
Acción del Actor	**Respuesta del Sistema**
1. La Audiencia accede a realizar un Ejercicio.	2. El sistema muestra una interfaz con los ejercicios.
3. La Audiencia selecciona un ejercicio a realizar.	4. El sistema muestra una interfaz con el e ejercicio seleccionado.
	5. El sistema muestra la misma interfaz con

	las clases predefinidas y etiquetadas.
	6. El sistema le permite colocar las tarjetas en clases predeterminadas.
7. La Audiencia coloca las clases en las categorías que él cree que corresponda y selecciona guardar y actualizar.	8. El sistema guarda y muestra una interfaz con las clases ubicadas por categorías.
9. La Audiencia solicita ver la solución del ejercicio.	10. El sistema muestra una interfaz con la solución del ejercicio resuelto.
11. La Audiencia revisa la solución y selecciona guardar.	12. El sistema guarda el ejercicio en el fichero XML de Ejercicios Realizados, finalizando así el Caso de Uso.
Sección "Resolver Ejercicio de Establecimiento de Clases Mixto"	
Acción del Actor	**Respuesta del Sistema**
1. La Audiencia accede a realizar un Ejercicio.	2. El sistema muestra una interfaz con los ejercicios, orientaciones, tarjetas y las clases predefinidas.
3. La Audiencia selecciona un ejercicio.	4. El sistema muestra en una interfaz un formulario con el ejercicio seleccionado.
	5. El sistema en la misma interfaz le permite colocar las tarjetas en las clases ya sean las predeterminadas o las creadas nuevas.
	6. El sistema le permite agregar clases a otras clases que ya están predefinidas.
7. La Audiencia coloca las tarjetas en las clases que crea conveniente.	
8. La Audiencia agrega un grupo de clases a otras que ya están predefinidas y selecciona guardar.	9. El sistema guarda y muestra un formulario con las clases agregadas a las que están predefinidas.
10. La Audiencia solicita ver la solución del	11. El sistema le muestra una interfaz con la

ejercicio.	solución del ejercicio.
12. La Audiencia revisa la solución y selecciona guardar.	13. El sistema guarda el ejercicio en el fichero XML de Ejercicios Realizados, finalizando así el Caso de Uso.
Poscondiciones	Queda resuelto el Ejercicio de Establecimiento de Clases**.**

Resolver Ejercicio de Establecimiento de Secuencia

Caso de Uso:	Resolver Ejercicio de Establecimiento de Secuencia
Actores:	Audiencia
Resumen:	El caso de Uso comienza cuando la Audiencia resuelve un ejercicio de Establecimiento de Secuencia, los guarda en el fichero XML de Ejercicios Realizados, finalizando así el Caso de Uso.
Precondiciones:	Deben estar preparados los ejercicios sobre los cuales se realizará el Ejercicio de Establecimiento de Secuencia.
Referencias	RF 11, RF 20, RF 21, RF 22,RF 23
Prioridad	Secundario
Flujo Normal de Eventos	
Sección "Inicial"	
Acción del Actor	**Respuesta del Sistema**
1. La Audiencia accede a realizar un ejercicio de establecimiento de secuencia.	2. El sistema le muestra una interfaz con los Ejercicios de Establecimiento de Clases resueltos.
3.La Audiencia selecciona un ejercicio	4. El sistema muestra una interfaz con las orientaciones y el ejercicio seleccionado.
	5. El sistema le permite ordenar las tarjetas dentro de las clases.
	6. El sistema le permite ordenar las clases.
7. La Audiencia accede a ordenar las tarjetas dentro de las clases.	
8. La Audiencia accede a ordenar las clases y	9. El sistema guarda las clases ordenadas.

selecciona guardar.	
10. La Audiencia solicita ver la sucesión de ordenación de las clases y las tarjetas.	11. El sistema muestra una interfaz con la secuencia de ordenación de las clases y las tarjetas.
12. La Audiencia revisa la ordenación y selecciona guardar.	13. El sistema guarda la ordenación en el fichero XML de Ejercicios Realizados, finalizando así el Caso de Uso.
Poscondiciones	Queda resuelto el Ejercicio de Establecimiento de Secuencia.

Procesar Resultados

Caso de Uso:	Procesar Resultados
Actores:	Arquitecto de Información
Resumen:	El Caso de uso comienza cuando el Arquitecto de Información procesa los resultados de los ejercicios de Card Sorting realizados por la Audiencia, le aplica el algoritmo de clústering, obtiene un Informe y lo guarda en el fichero XML de Resultados Finales, finalizando así el Caso de Uso.
Precondiciones:	Deben estar guardados los ejercicios realizados por la Audiencia.
Referencias	RF 26,RF 27, RF 28, RF29, RF 30, RF 31
Prioridad	Critica

Flujo Normal de Eventos	
Sección "Inicial"	
Acción del Actor	**Respuesta del Sistema**
1. El Arquitecto accede a procesar los resultados de los ejercicios realizados por la Audiencia.	2. El sistema le muestra la plantilla de resultados de los ejercicios.
3. El Arquitecto procesa los resultados aplicándole algoritmos de clústering y selecciona guardar.	4. El sistema muestra una interfaz con los resultados procesados e informa que se guardaron en el fichero XML de Resultados Finales.
5. El Arquitecto solicita un informe con los	6. El sistema genera un informe que contiene

resultados de los ejercicios.	los resultados del procesamiento de las respuestas generadas por la Audiencia.
	7. El sistema muestra en una interfaz el informe de los resultados antes procesados dando la opción de imprimirlo.
8. El Arquitecto selecciona imprimir el informe.	9. El sistema ejecuta la acción de imprimir
Poscondiciones	Queda generado e impreso un informe sobre los resultados de Ejercicios de Card Sorting resueltos por la Audiencia.

Conclusiones

Una vez concluido el Capítulo, se cumplieron con los objetivos del mismo, pues se realizó el modelo de negocio de los procesos referente a la técnica. Además se obtuvieron los diferentes diagramas concebidos según la metodología RUP, para modelar esos procesos, la interacción entre los actores y casos de uso, el modelo de objeto con los trabajadores y las clases entidades con las que se relacionan. Se realizó el diagrama de Casos de Uso del Negocio y se describieron los casos de usos representados, se desarrollo con éxito la propuesta de solución del sistema a partir del análisis de los procesos del sistema. Se representaron los requisitos funcionales y no funcionales con que contará el software, al igual que los procesos existentes en el sistema a través de un Diagrama de Casos de Uso del Sistema; se describieron paso a paso todas las acciones de los actores del sistema con los casos de uso que interactuaban, dando paso al análisis y factibilidad del sistema.

CAPÍTULO 3: ANÁLISIS Y RESULTADOS DE LA FACTIBILIDAD DEL SISTEMA

Introducción.

En este capítulo se modela los diagramas de clases del análisis correspondiente; donde se aprecia la interacción del usuario con las diferentes interfaces del sistema y las clases entidades involucradas, además se abordarán aspectos relacionados con la estimación de esfuerzos de desarrollo del sistema, utilizando como variante para la estimación el Análisis de Puntos de Casos de Uso y se realizará una valoración sobre el resultado obtenido de la estimación.

3.1 Análisis del Sistema

El análisis del sistema es uno de los flujos de trabajos realizados durante el proceso del software, este se desarrolla fundamentalmente dentro de la fase de elaboración. El análisis consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. [23]

A continuación se nombran los tipos de clases que son utilizados en el modelo de análisis:

CI_<Nombre de la clase>: estas clases modelan la interacción entre los usuarios y el sistema, es decir, ventanas, formularios, dispositivos, sistemas externos, etc. Ejemplo CI_Gestionar_Proyecto.

CC_<Nombre de la clase>: estas clases encapsulan el comportamiento de cada caso de uso y coordinan el trabajo de las clases interfaz y entidad. Ejemplo CC_Ejercicio Clases.

CE_<Nombre de la clase>: estas clases modelan toda la información del sistema que posee una vida larga y que puede ser persistente. Ejemplo CE_FicheroXML_Ejerc_Prep.

A continuación se presentan los diagramas de clases de los casos de uso descritos en el documento:

Diagrama de Clases de Análisis Gestionar Proyecto de Card Sorting

[23] Op. Cit(19)

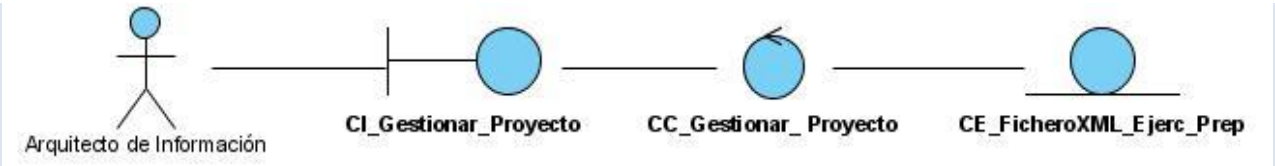


Diagrama de Clases de Análisis Preparar Ejercicio de Establecimiento de Clases

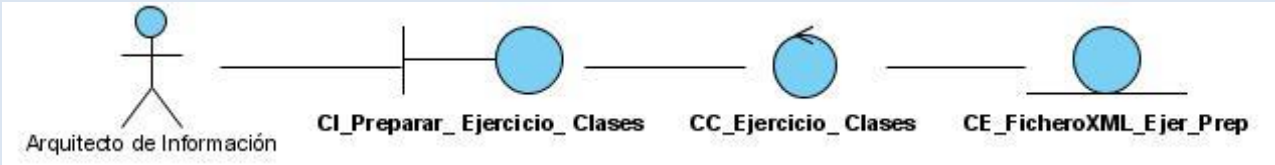


Diagrama de Clases de Análisis Preparar Ejercicio Establecimiento de Bloques

Diagrama de Clases de Análisis Preparar Ejercicio de Establecimiento de Secuencia

Diagrama de Clases de Análisis Resolver Ejercicio de Establecimiento de Clases

Diagrama de Clases de Análisis Resolver Ejercicio de Establecimiento de Bloques

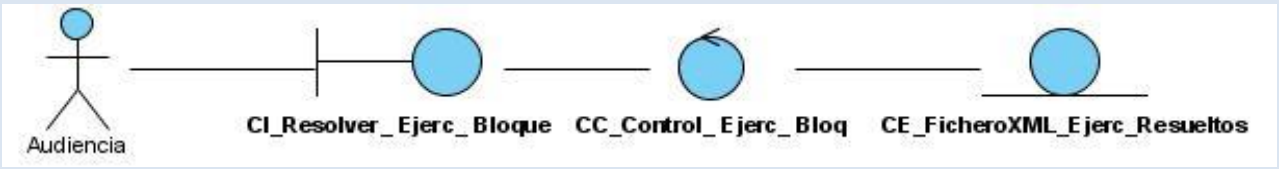
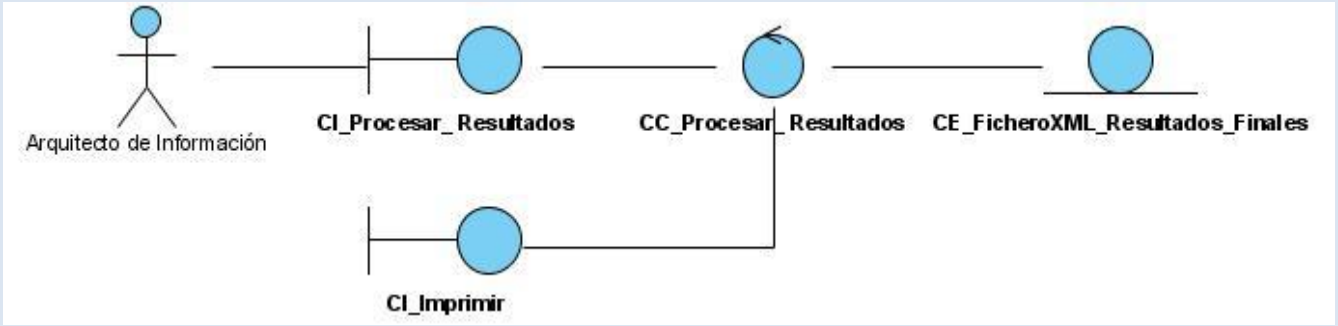


Diagrama de Clases de Análisis Resolver Ejercicio de Establecimiento de Secuencia

Diagrama de Clases de Análisis Procesar Resultados



3.1.1 Diagramas de colaboración

Diagrama de Colaboración Modificar Proyecto

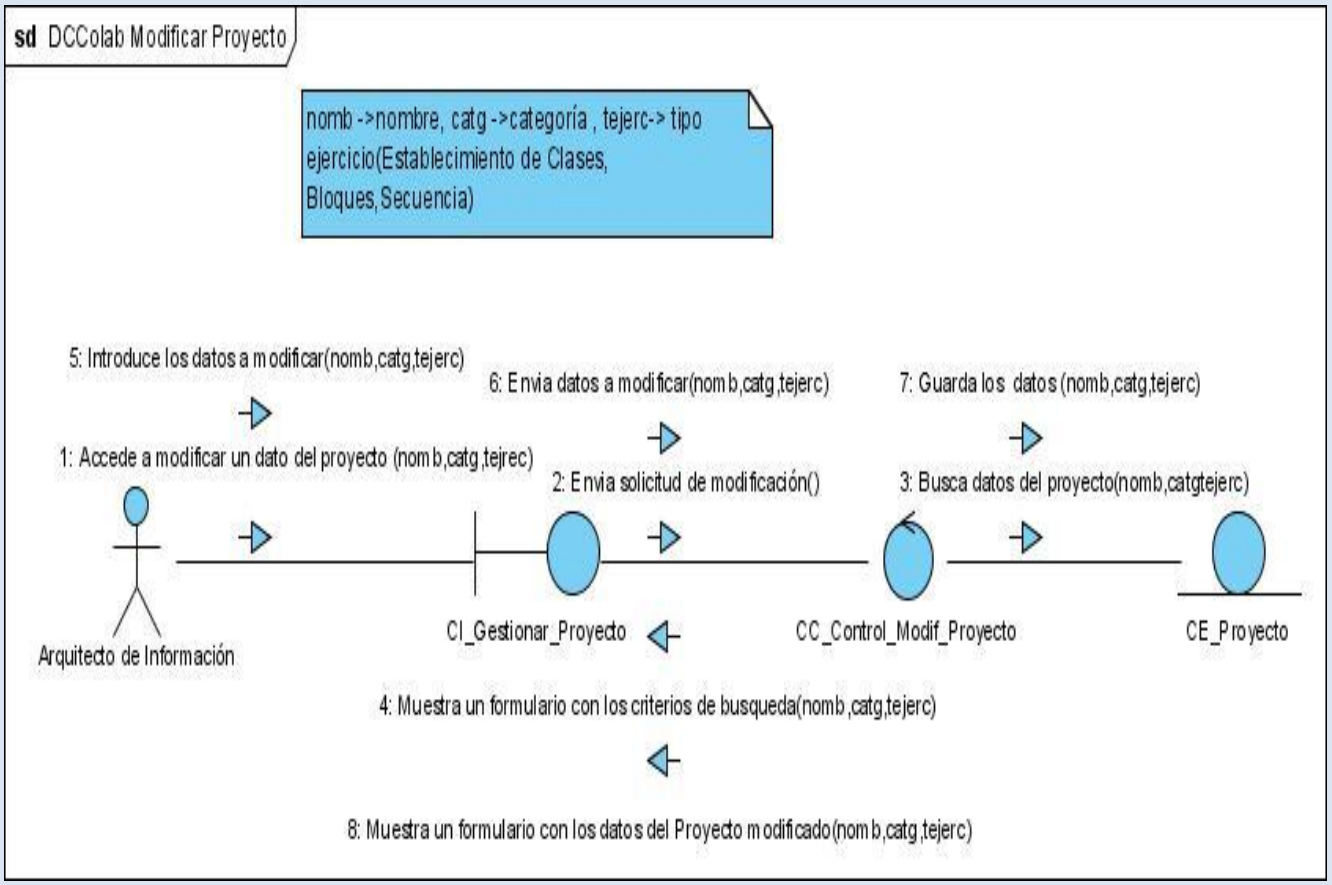


Diagrama de Colaboración Crear Proyecto

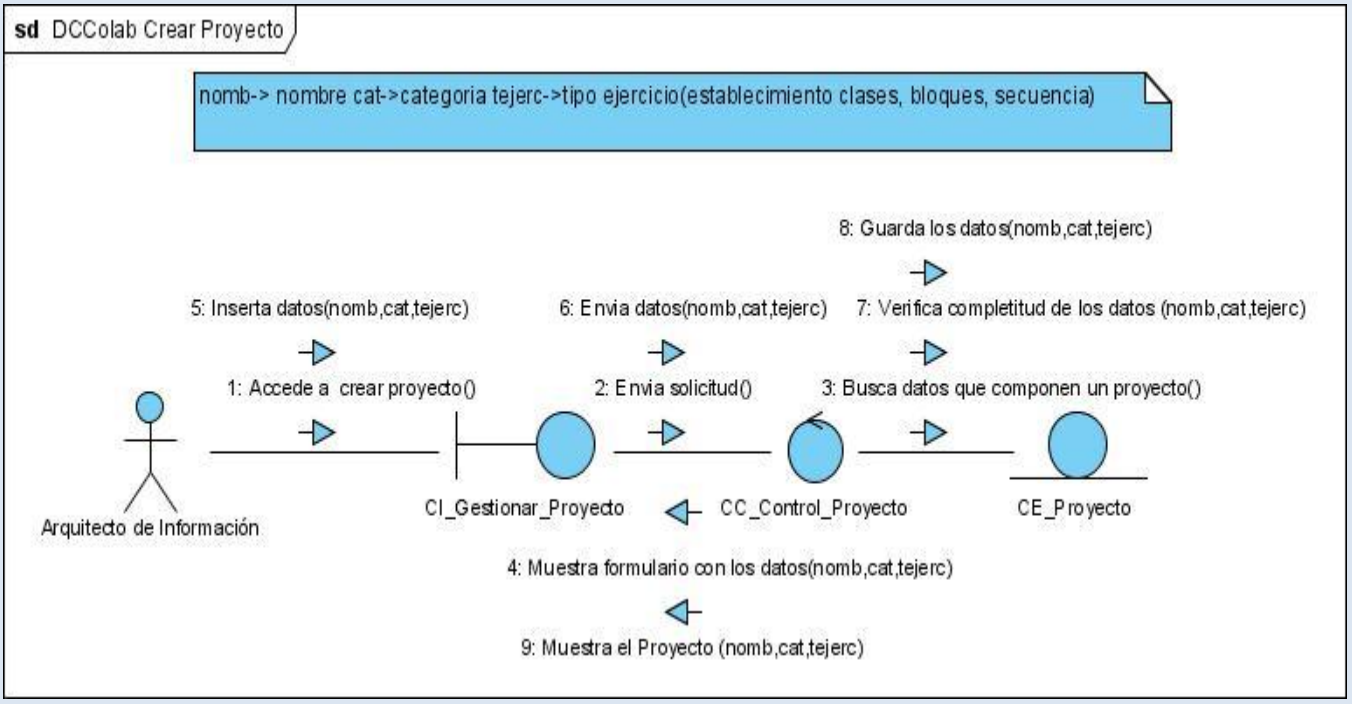


Diagrama de Colaboración Preparar Ejercicio Clases

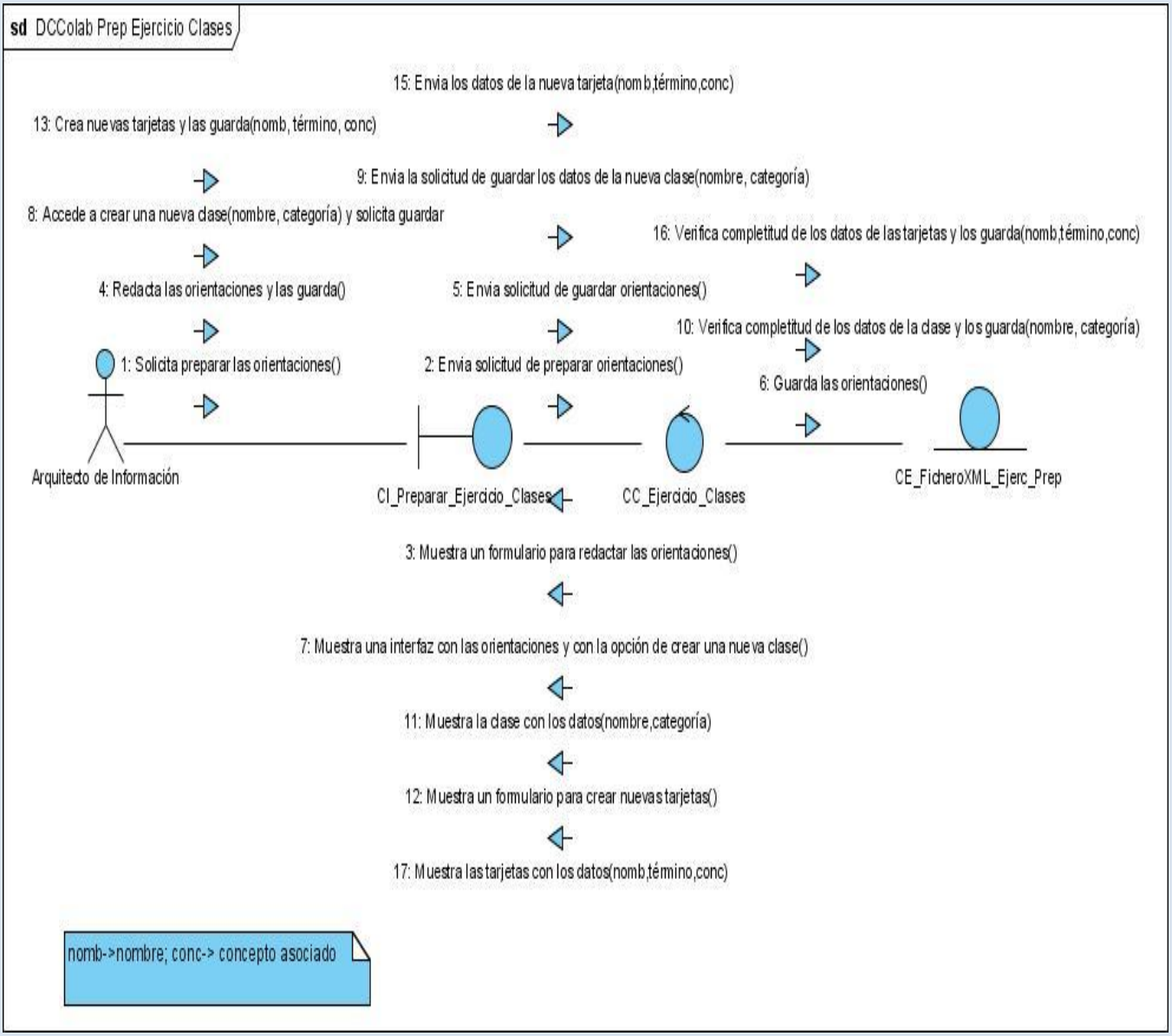


Diagrama de Colaboración Preparar Ejercicio de Bloques

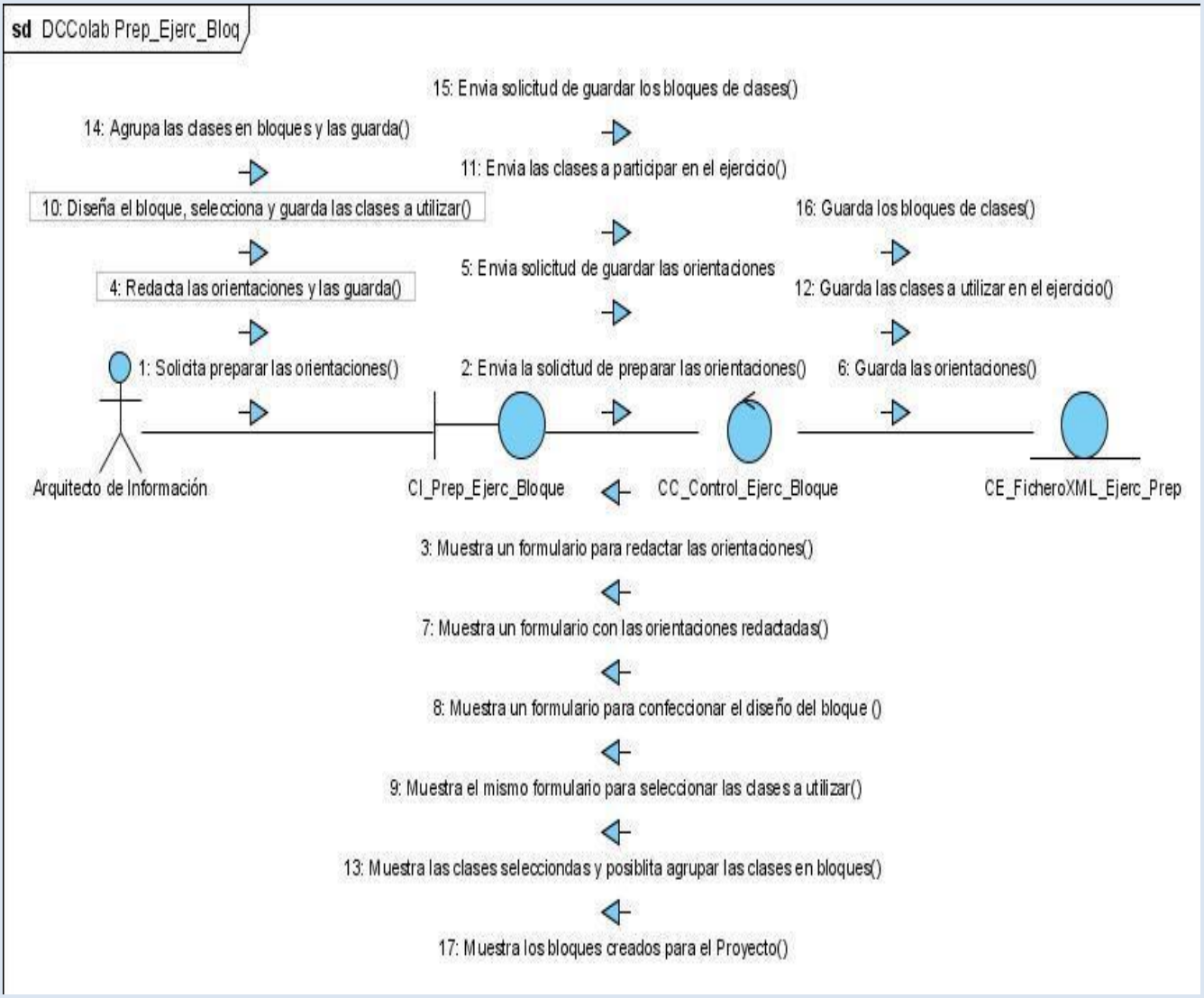


Diagrama de Colaboración Preparar Ejercicio Secuencia

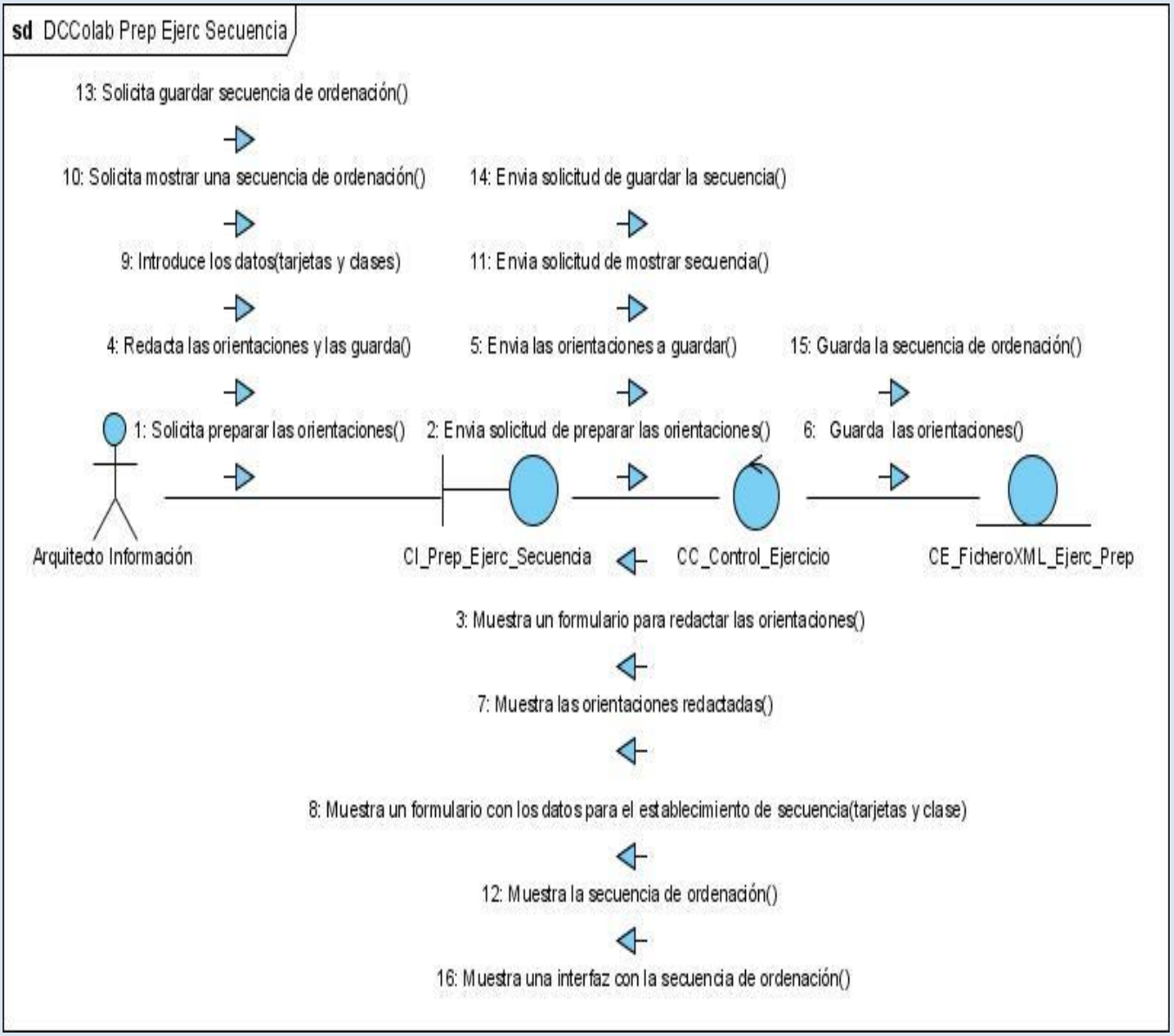


Diagrama de Colaboración Resolver Ejercicio Establecimiento Clase Abierto

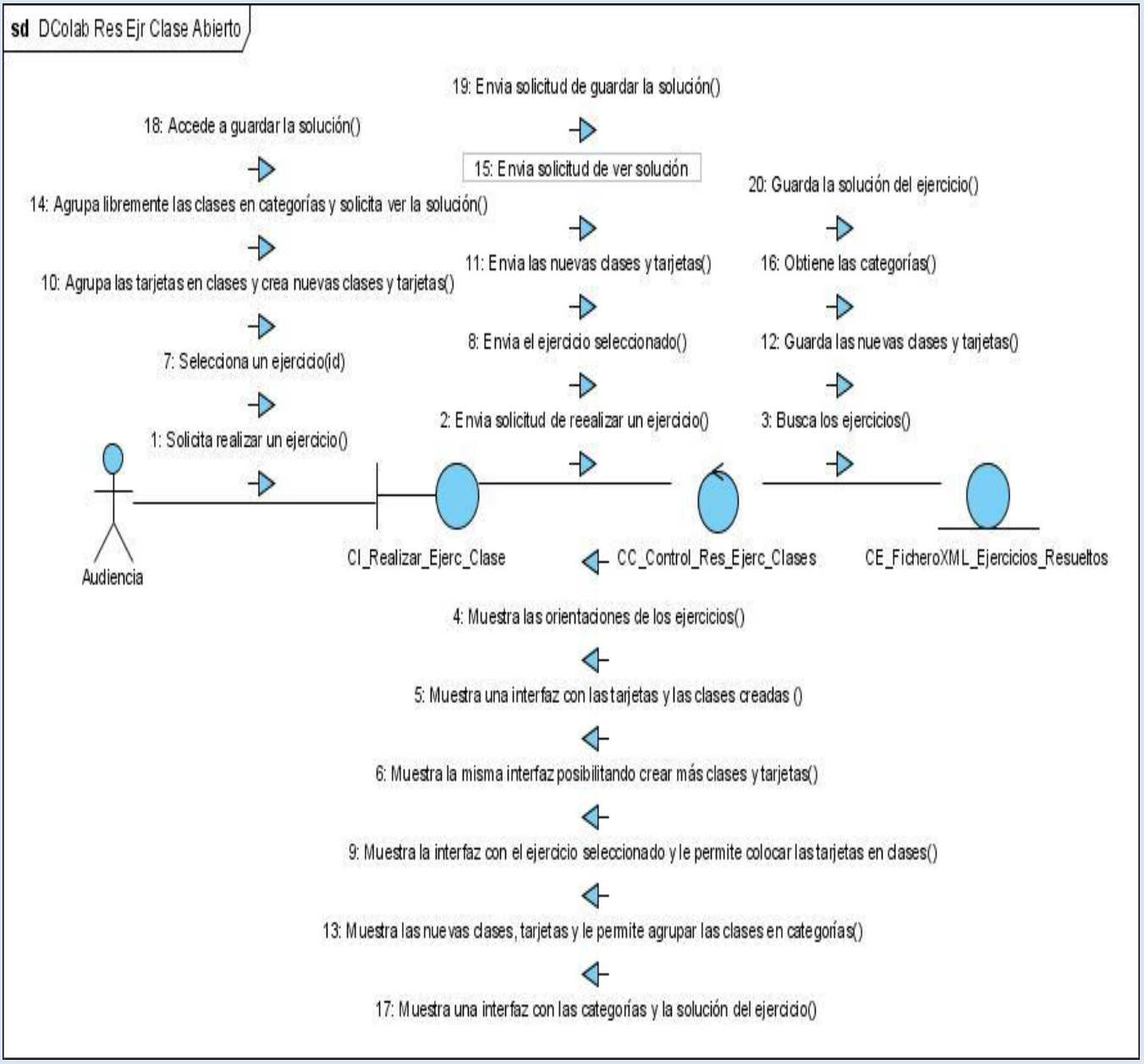


Diagrama de Colaboración Resolver Ejercicio de Establecimiento Clases Cerrado

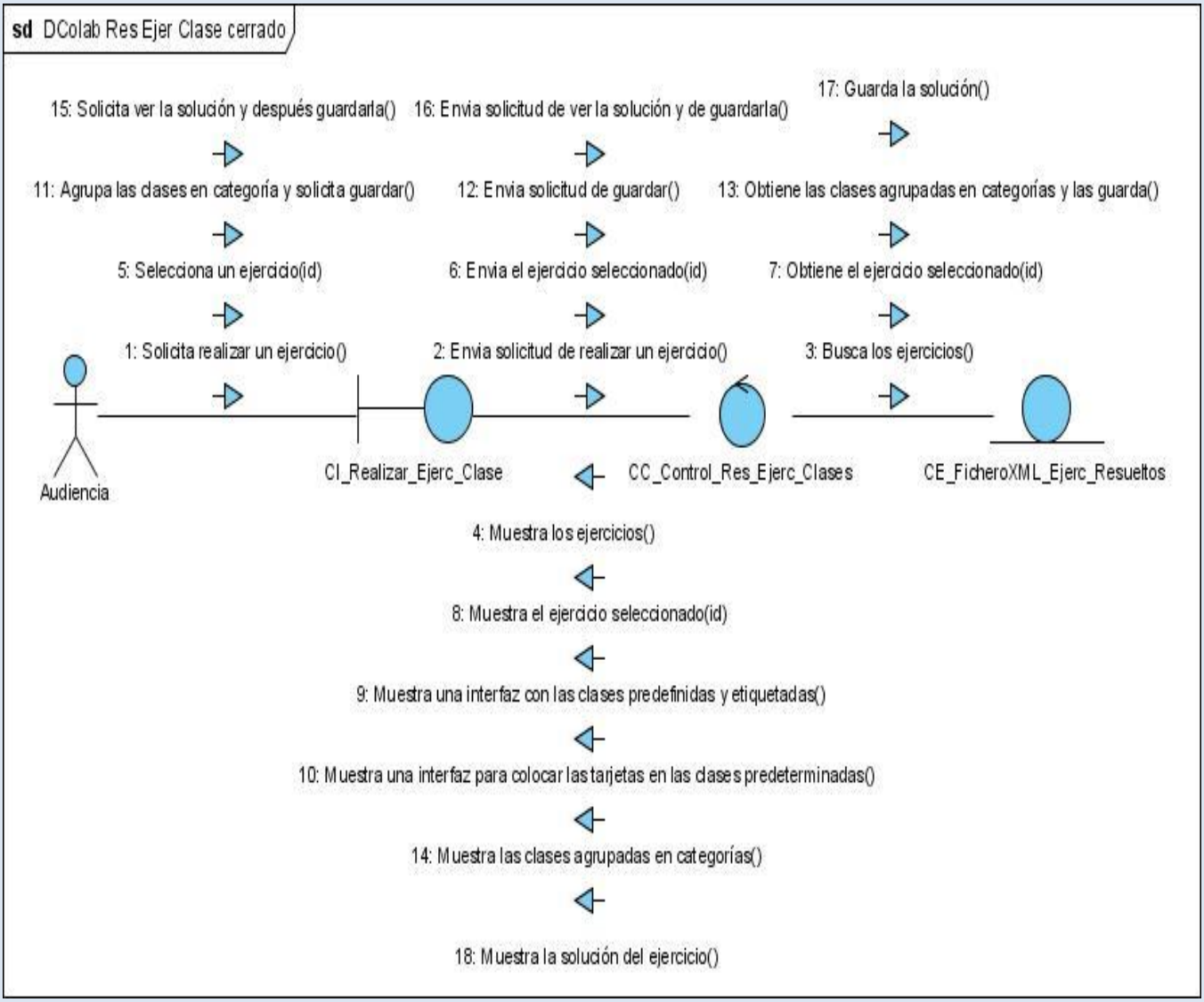


Diagrama de Colaboración Resolver Ejercicio de Establecimiento de Clase Mixta

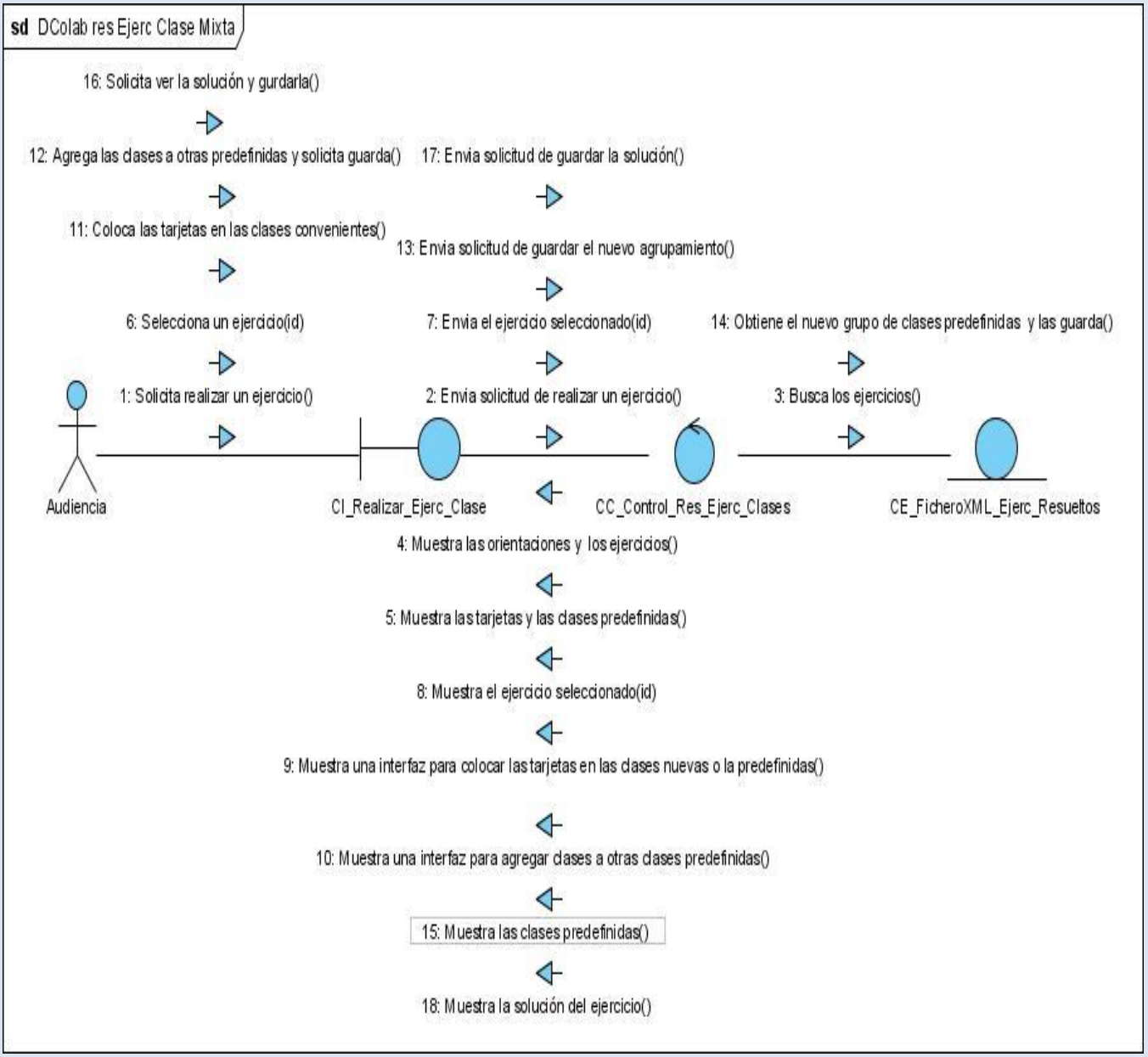


Diagrama de Colaboración Resolver Ejercicio Bloques

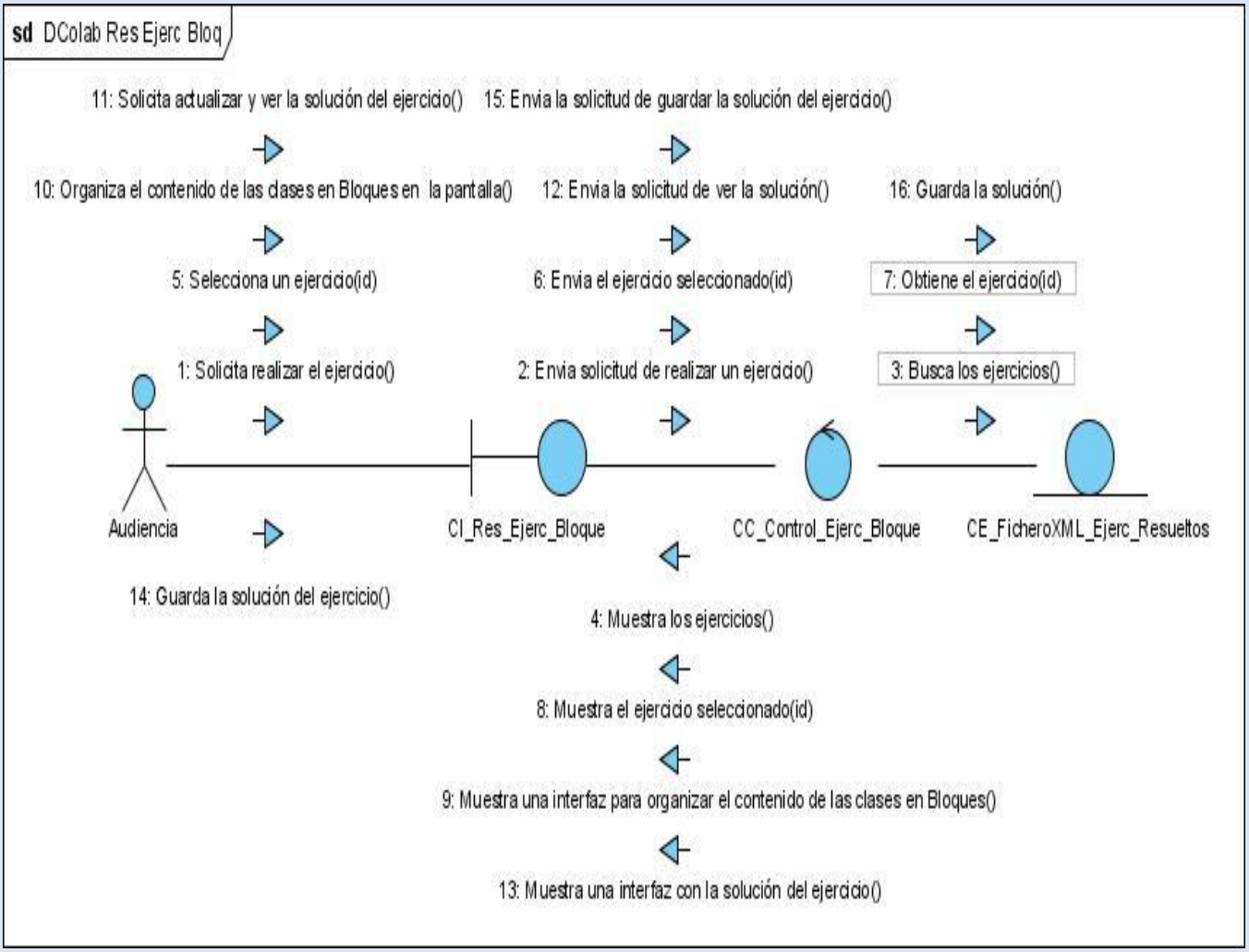


Diagrama de Colaboración Resolver Ejercicio Secuencia

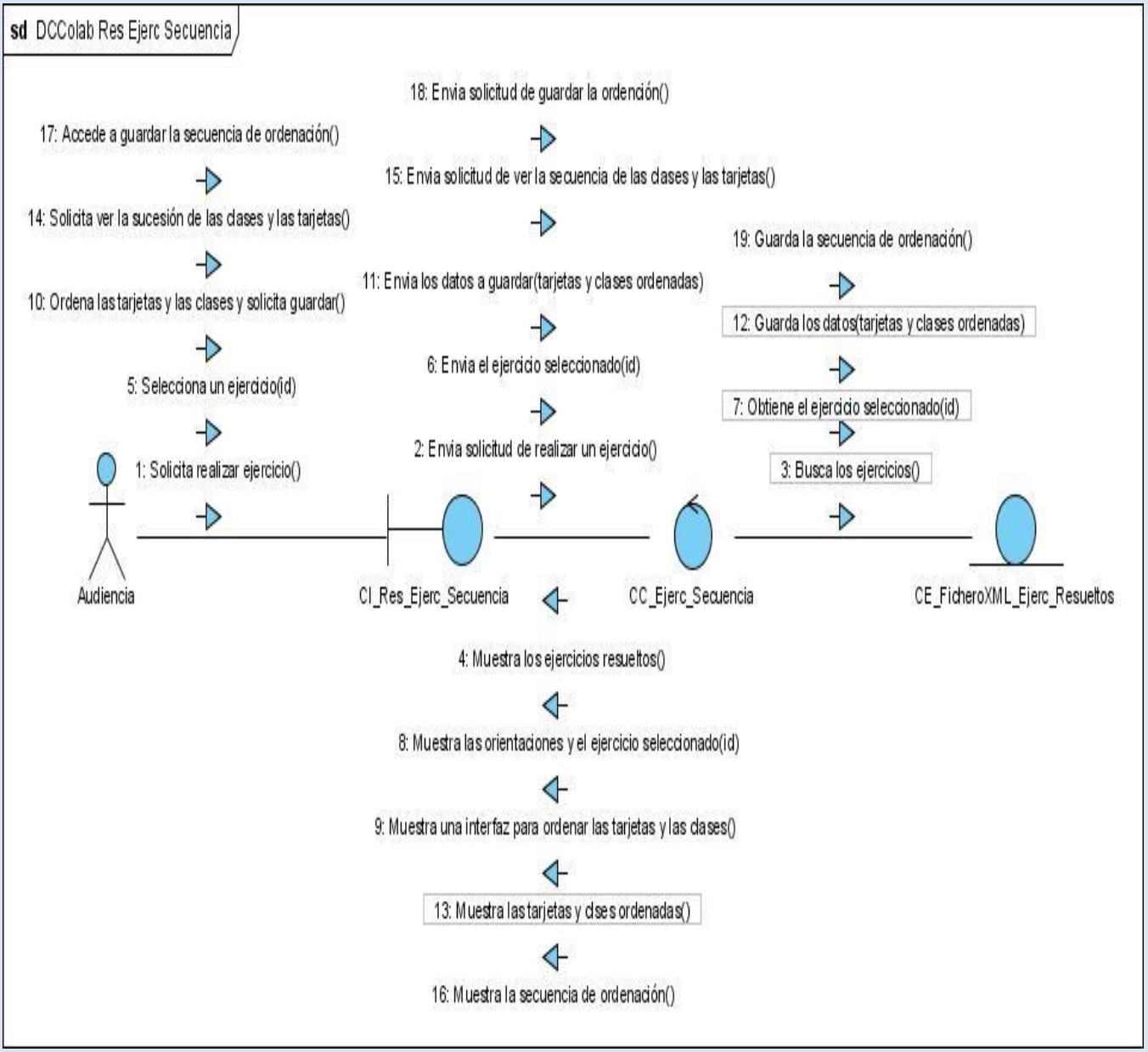
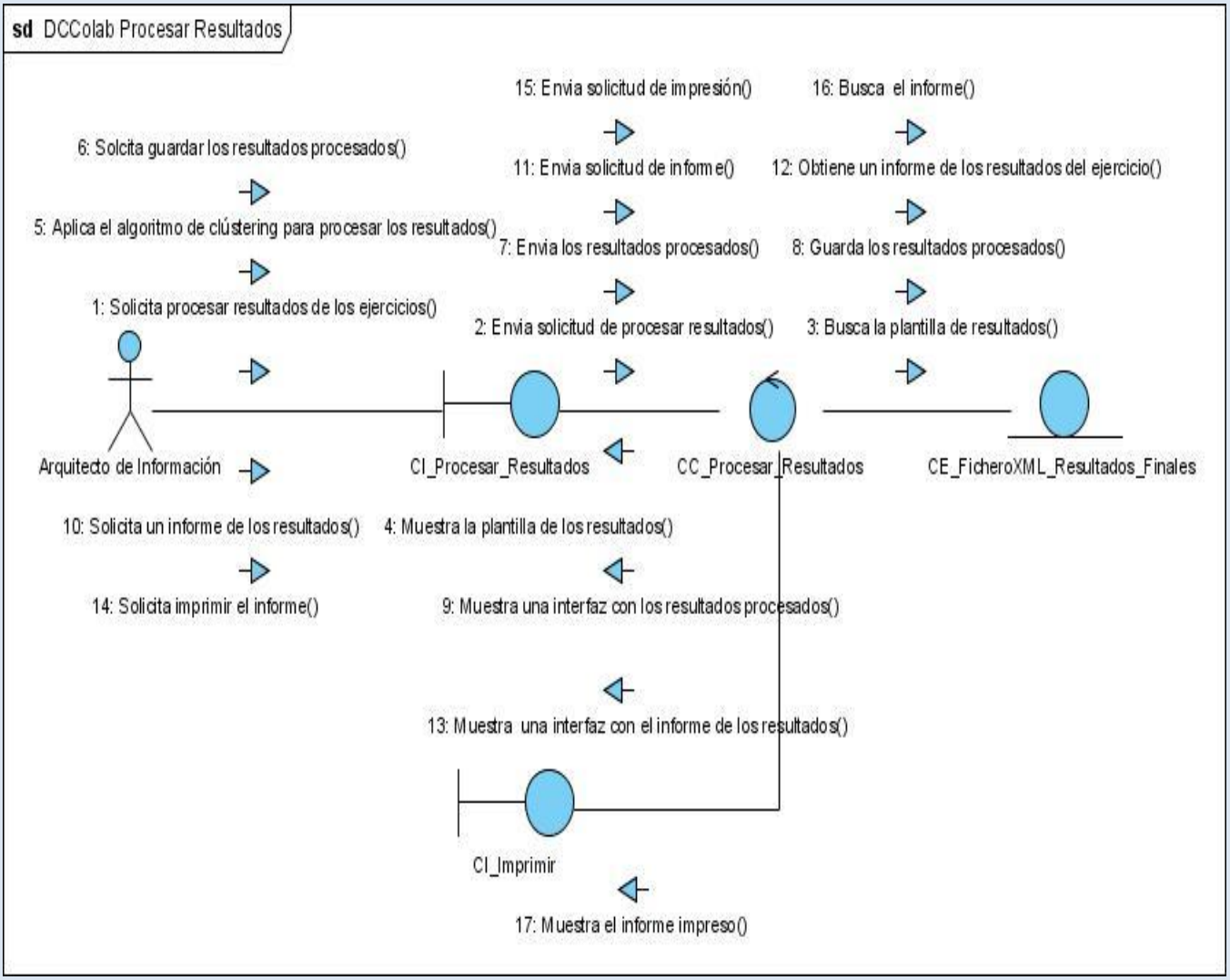


Diagrama de Colaboración Procesar Resultados



3.2 Análisis de la factibilidad del sistema

Siempre que se planifique un proyecto se tienen que obtener estimaciones del costo y esfuerzo humano requerido por medio de las mediciones de software que se utilizan para recolectar los datos cualitativos acerca del software y sus procesos para aumentar su calidad. El objetivo fundamental de la planificación y el análisis de la factibilidad es establecer planes razonables para desarrollar la

Ingeniería de Software y manejar los cambios de los proyectos de Software incluyendo la actividad de estimar los resultados del proyecto y los valores de costo, tiempo y recursos requeridos, además establece los pasos necesarios y define el plan de desarrollo. La planificación se logra mediante un procesamiento de la información que lleve a estimaciones razonables. [24]

En el presente capítulo se abordarán aspectos relacionados con la estimación de esfuerzos de desarrollo del sistema, utilizando como variante para la estimación el Análisis de Puntos de Casos de Uso y como conclusión se realizará una valoración sobre el resultado obtenido de la estimación.

3.3 Planificación basada en casos de uso.

Para lograr una planificación se requiere de técnicas de estimación, esto quiere decir que a partir de determinados parámetros se predeterminan variables a estimar como el costo, el esfuerzo y el tiempo necesarios para obtener el software. Donde el esfuerzo se traduce al total de tiempo que gasta una persona trabajando en el desarrollo del proyecto de software (horas/persona | mes/persona).

La planificación basada en casos de uso es un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores. A continuación, se detallan los pasos a seguir para la aplicación de éste método que son: Cálculo de Puntos de Casos de Uso sin ajustar (UUCP), Factor de Peso de los Actores sin ajustar (UAW), Factor de Peso de los Casos de Uso sin ajustar (UUCW) y Cálculo de Puntos de Casos de Uso ajustados (UCP). [25]

Paso 1. Identificar los Puntos de casos de uso Desajustados

UUCP = UAW + UUCW

Donde:

UUCP: Puntos de Casos de Uso sin ajustar

UAW: Factor de Peso de los Actores sin ajustar

[24] GIRALDO, O. P. Métricas, Estimación y Planificación en Proyectos de Software., 2007. [Disponible en]: http://www.willydev.net/Descargas/WillyDEV_PlaneaSoftware.Pdf

[25] CAPIS, C. D. I. D. S. E. I. D. C. ESTIMACIÓN DEL ESFUERZO BASADA EN CASOS DE USO, 2007]. [Disponible en]: http://www.itba.edu.ar/capis/rtis/rtis-6-1/estimacion-del-esfuerzo-basada-en-casos-de-usos.pdf

UUCW: Factor de Peso de los Casos de Uso sin ajustar

El valor del Factor de Peso de los Actores sin ajustar se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos como se muestra a continuación.

Tipo	**Descripción**	**Peso**	**Cant* peso**
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface)	1	0*1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2	0*2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3	2*3
		UAW	**6**

El valor del Factor de Peso de los Casos de Uso sin ajustar se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción se entiende como una secuencia de actividades atómica, es decir, se efectúa la secuencia de actividades completa, o no se efectúa ninguna de las actividades de la secuencia. Los criterios se muestran en la siguiente tabla:

Para calcular el Factor de Peso de los Casos de Uso sin ajustar se tienen en cuenta las cantidades de transacciones de los casos de uso del sistema, donde una transacción es

Tipo	**Descripción**	**Peso**	**Cant* peso**
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5	6*5=30
Medio	El Caso de Uso contiene de 4 a 7 transacciones	10	1*10=10
Complejo	El Caso de Uso contiene más de 8 transacciones	15	1*15=15
		UUCW	**55**

Luego:

UUCP = UAW + UUCW = 6 + 55 **UUCP = 61**

Paso 2. Ajustar los Puntos de casos de uso

UCP = UUCP * TCF * EF

Donde:

UCP: Puntos de Casos de Uso ajustados

UUCP: Puntos de Casos de Uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor de ambiente

El coeficiente de **Factor de complejidad Técnica (TCF)** se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante. A continuación se muestra el significado y el peso de cada uno de éstos factores:

TCF = 0.6 + 0.01 * Σ **(Peso$_i$** * **Valor$_i$)** (Donde Valor es un número del 0 al 5)

Significado de los valores

0: No presente o sin influencia

1: Influencia incidental o presencia incidental

2: Influencia moderada o presencia moderada

3: Influencia media o presencia media

4: Influencia significativa o presencia significativa

5: Fuerte influencia o fuerte presencia

Factor	Descripción	Peso	Valor	Σ (Peso$_i$ * Valor$_i$)
T1	Sistema distribuido	2	0	0
T2	Objetivos de performance o	1	4	4

	tiempo de respuesta			
T3	Eficiencia del usuario final	1	2	2
T4	Procesamiento interno complejo	1	3	3
T5	El código debe ser reutilizable	1	2	2
T6	Facilidad de instalación	0.5	3	1.5
T7	Facilidad de uso	0.5	3	1.5
T8	Portabilidad	2	3	6
T9	Facilidad de cambio	1	3	3
T10	Concurrencia	1	3	3
T11	Incluye objetivos especiales de seguridad	1	1	1
T12	Provee acceso directo a terceras partes	1	2	2
T13	Se requieren facilidades especiales de entrenamiento a los usuarios.	1	3	3
			Total	**32**

Comentarios:

T1: El sistema no es distribuido ya que es una aplicación de escritorio.

T2: Debe tener un tiempo de respuesta significativa debido a que será una aplicación que se trabajará sobre la misma PC.

T3: El usuario no requiere mucha preparación para usar el sistema.

T4: Este sistema requiere de algoritmos de clústering para su implementación y así obtener un mejor procesamiento de los resultados de los ejercicios.

T5: El código fuente no se requiere que sea reutilizable para otras implementaciones pero puede serlo en algún caso específico.

T6: La instalación es moderada.

T7: El sistema tendrá una interfaz amigable y fácil de usar.

T8: Será una aplicación con una gran capacidad para ser ejecutada en diferentes sistemas informáticos en el ámbito de la arquitectura de información.

T9: Estará disponible a realizarle los cambios necesarios.

T10: La concurrencia tendrá una presencia media permitiendo que los usuarios asuman que la aplicación se ejecuta atómicamente, como si no existieran otras aplicaciones ejecutándose concurrentemente.
T11: No se necesita de seguridad por parte de los usuarios.
T12: No necesariamente posee accedo directo a terceras áreas.
T13: Su fácil uso permite que el personal no tenga que alcanzar un alto nivel de capacitación.

TCF = 0.6 + 0.01 * **32**
TCF = 0.92

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos factores son los que se contemplan en el cálculo del **Factor de Ambiente (EF)**. El cálculo del mismo es similar al cálculo del Factor de complejidad técnica, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5.
En la siguiente tabla se muestra el significado y el peso de cada uno de éstos factores.

EF = 1.4 - 0.03 * Σ **($Peso_i$** * **$Valor_i$)** (Donde Valor es un número del 0 al 5)

Factor	Descripción	Peso	Valor	Σ ($Peso_i$ * $Valor_i$)
E1	Familiaridad con el modelo de proyecto utilizado	1.5	2	3
E2	Experiencia en la aplicación	0.5	1	0.5
E3	Experiencia en orientación a objetos	1	3	3
E4	Capacidad del analista líder	0.5	3	1.5
E5	Motivación	1	3	3
E6	Estabilidad de los requerimientos	2	2	4
E7	Personal part-time	-1	3	-3
E8	Dificultad del lenguaje de programación	-1	3	-3
			Total	**9**

Comentarios:
E1: El equipo de trabajo está un poco familiarizado con el modelo utilizado.

E2: Nunca el equipo ha trabajado en aplicaciones con estas características.

E3: El equipo tiene una experiencia normal en la programación orientada a objetos.

E4: El analista que ha desarrollado el sistema tiene una capacidad asequible para lograr un buen producto final.

E5: Existe una motivación media por parte del equipo.

E6: Los requerimientos han sufrido algunos cambios desde su levantamiento.

E7: El equipo no trabaja a tiempo completo ya que tienen otras responsabilidades.

E8: La complejidad del lenguaje de programación es media (C#).

EF = 1.4 - 0.03 * 9

EF = 1.13

Luego: UCP = UUCP * TCF * EF= 61 * 0.92 * 1.13 **UCP= 63.41**

Paso 3. Calcular esfuerzo de FT Implementación

E= UCP * CF

Donde:

E: esfuerzo estimado en horas-hombre

UCP: Puntos de Casos de Uso ajustados

CF: Factor de conversión

El **Factor de Conversión (CF)** se calcula a continuación:

De los Puntos de Casos de Uso a la estimación del esfuerzo se contabilizan cuántos factores de los que afectan al Factor de ambiente están por debajo del valor medio (3), para los factores del E1 a E6 y cuántos están por encima del valor medio (3), para los factores E7 y E8.

CF = 20 horas-hombre (si $Total_{EF} \leq 2$)

CF = 28 horas-hombre (si $Total_{EF} = 3$ ó $Total_{EF} = 4$)

CF = abandonar o cambiar proyecto (si $Total_{EF} \geq 5$)

$Total_{EF}$ = Cant EF < 3 (entre E1 –E6) + Cant EF > 3 (entre E7, E8)

Como $Total_{EF} = 2 + 0$

$Total_{EF} = 2$

CF = 20 horas-hombre (porque Total $_{EF}$ = 2)

Luego E= UCP * CF

E = 63.41 * 20 horas-hombre

E ≈ 1268 horas-hombre

Paso 4. Calcular esfuerzo de todo el proyecto

Actividad	**% esfuerzo**	**Valor esfuerzo**
Análisis	10%	317 horas-hombre
Diseño	20%	634 horas-hombre
Implementación	40%	1268 horas-hombre
Prueba	15%	475.5 horas-hombre
Sobrecarga	15%	475.5 horas-hombre
Total	100%	3170 horas-hombre

Como el valor de esfuerzo calculado representa el esfuerzo del Flujo de Trabajo implementación, por comparación salen el resto de los esfuerzos y la suma de ellos es el **esfuerzo total (E_T)**. Para la realización del proyecto se trabaja diario 5 horas, 6 días a la semana y un mes tiene como promedio 4 semanas se trabajan 24 días; la cantidad de horas que puede trabajar una persona mensual es de 100 horas. Si **E_T = 3170 horas-hombre** y por cada 120 horas se tiene 1 mes eso daría un **E_T = 26.4 mes-hombre.** Esto significa que el equipo de trabajo del proyecto de 6 personas puede realizar el trabajo completo del sistema en más o menos 4.4 meses, aproximadamente 4 meses.

3.3 Beneficios tangibles e intangibles

Con la realización del análisis del sistema se le proporcionarán varios **beneficios tangibles** para el Departamento de Arquitectura de Información, primeramente los procesos que se desarrollan en el departamento podrán controlarse y gestionarse con mejores resultados, el acceso para manipular la organización de la información que se maneja será con mayor puntualidad que en el pasado. Los empleados del departamento utilizando las funcionalidades que proporciona este sistema lograrán reducir mucho más el tiempo requerido para la realización de sus tareas y con una mayor calidad. Hay

que decir que los beneficios a que se hace referencia no están comprobados ya que por ahora el sistema está sólo en la fase de análisis. Al implementarse y desplegarse el sistema se podrán monitorear estos efectos y los estimados se tornarán en valores reales.

Dentro de los **beneficios intangibles** que proporciona el sistema se encuentra que el incremento de la satisfacción de los empleados será mayor ya que se eliminarán tareas de naturaleza tediosa como distribuir tarjetas con las diferentes temáticas a los clientes para así obtener el resultado de la web a diseñar, tarea que será sustituida por las búsquedas automatizadas de las diferentes categoría y temáticas. El sistema traerá aparejado el incremento de la capacidad organizativa del Departamento y sobre todo el aumento de la calidad y rapidez del diseño de la web centrado en el usuario.

3.4 Análisis de costos y beneficios.

Después de haber realizado la estimación del esfuerzo para llevar a cabo el desarrollo del sistema y el análisis de los beneficios tangibles e intangibles expuestos anteriormente, cabe señalar el ahorro que trae como beneficio, pues el sistema está siendo desarrollado por estudiantes sin que se incurra en gastos de salario, además los gastos de recursos son mínimos ya que será desarrollado en su totalidad por tecnologías de licencias de software libre, proporcionando sin dudas un ahorro significativo en compras de licencias propietarias. Se puede plantear, por los beneficios tangibles e intangibles que fueron mencionados anteriormente, el producto es viable y justificado, reportando grandes beneficios al país y siendo un aporte más para la informatización de la sociedad cubana y en particular a la Universidad de las Ciencias Informáticas (UCI).

Conclusiones

En este capítulo se realizó el análisis del sistema y el estudio de factibilidad, obteniéndose como resultado la estimación un tiempo de aproximadamente 4 meses para el desarrollo del sistema con 6 hombres, llegando a la conclusión que es factible implementar el sistema propuesto, argumentado por los beneficios tangibles e intangibles que este ofrece.

CONCLUSIONES

- Se realizó un estudio detallado del funcionamiento de la técnica Card Sorting u Ordenación de Tarjetas.
- Se investigó sobre las herramientas y tecnologías actuales, principalmente las relacionadas con software libre.
- Se seleccionó como herramienta CASE el Visual Paradigm basado en el lenguaje UML, utilizando la metodología RUP.
- Se investigó sobre los sistemas existentes tanto a nivel nacional como internacional que apliquen el proceso de dicha técnica.
- Se realizó un levantamiento de requisitos que permitió conocer a fondo las necesidades de una herramienta informática que aplique el mecanismo.
- Se realizó el análisis del sistema cumpliendo con los requisitos necesarios obtenidos en la captura de los mismos.
- Se obtuvo la modelación de un sistema automatizado centrado en el usuario a través de la técnica del Card Sorting.

RECOMENDACIONES

- Continuar con la investigación para garantizar el desarrollo de una futura versión del sistema.
- Diseñar e implementar completamente la solución propuesta, así se desarrollaría un producto que responda a las exigencias planteadas por el cliente, el cual permita gestionar la información de forma rápida y segura.
- Desarrollar pruebas planificadas para comprobar que la aplicación cumplirá con las funcionalidades requeridas.
- Extender el software una vez implementado por el resto del país, así se lograría el diseño web centrado en el usuario al 100 %.

BIBLIOGRAFÍA

1. **Berzal Galiano, Fernando.** Curso de C# . *Curso de C# .* [Online] [Citado: mayo 5, 2008.]

2. **Carlos Garcia, Juan.** Card Sorting. El medio es el msje . *Card Sorting. El medio es el msje .* [Online] [Citado: mayo 4 , 2008.] http://usalo.es/63/card-sorting-el-medio-es-el-mensaje/.

3. **Carreras Plaza, Jesús y Guaderrama Hernández, Maritza.** *El Enfoque Cualitativo en el desarrollo de Arquitecturas de Información: Card Sorting + Entrevista Abierta***2006.** Extreme Programming:A gentle introduction. *Extreme Programming:A gentle introduction.* [Online] febrero 17, 2006. [Citado: mayo 7, 2008.] http://www.extremeprogramming.org/.

4. **Carreras, Jesús. 2007.** [cadius] Fwd: Ayuda urgente con CardSorting, concretamente con CardCluster . *[cadius] Fwd: Ayuda urgente con CardSorting, concretamente con CardCluster .* [Online] septiembre 17, 2007. [Citado: mayo 3, 2008.] http://www.cadius.org/pipermail/lista_cadius.org/2007-September/005511.html.

5. **Catuxa. 2005.** Card Sorting en la practica. *Card Sorting en la practica.* [Online] mayo 5, 2005. [Citado: mayo 20, 2008.] http://www.deakialli.com/2005/05/05/card-sorting-en-la-practica/.

6. Conduct card-sorting studies online. *Conduct card-sorting studies online.* [Online] [Citado: mayo 3, 2008.] http://websort.net/.

7. Development... *Development...* [Online] mayo 22, 2005. [Citado: mayo 3, 2008.] http://cardsword.sourceforge.net/.

8. Ejemplo de desarrollo software utilizando la metodología XP. *Ejemplo de desarrollo software utilizando la metodología XP.* [Online] enero 9, 2004. [Citado: mayo 7, 2008.] http://www.dsic.upv.es/asignaturas/facultad/lsi/ejemploxp/.

9. Ejemplos java y C/linux. *Ejemplos java y C/linux.* [Online] [Citado: mayo 7, 2008.] http://www.chuidiang.com/ood/metodologia/scrum.php.

10. **Fernández González, Jorge. 2007.** ¿SCRUM para DWH? *¿SCRUM para DWH?* [Online] enero 23, 2007. [Citado: mayo 7, 2008.] http://sistemasdecisionales.blogspot.com/2007/01/scrum-para-dwh.html.

11. **Fowler, Martin.** La Nueva Metodología. *La Nueva Metodología.* [Online] [Citado: mayo 7, 2008.] http://www.programacion.com/tutorial/nuevametodologia/5/.

12. glosario:extreme_programming. *glosario:extreme_programming.* [Online] [Citado: mayo 7, 2008.] http://www.planetacodigo.com/wiki/glosario:extreme_programming.

13. **Gutierrez, Luis, Miguelez, Xavier and Ruiz, Inmaculada.** COLABORACIÓN INTER-EUROPEA EN EL MARCO DEL PROYECTO CANDLE. *COLABORACIÓN INTER-EUROPEA EN EL MARCO DEL PROYECTO CANDLE.* [Online] [Citado: mayo 4, 2008.] http://giac.upc.es/PAG/giac_cas/giac_jac/03/Candle(formAT3).htm.

14. **Hassan Montero, Yusef and Martín Fernández, Francisco J. 2004.** Estructuración de la Información: Aproximación descendente. *Estructuración de la Información: Aproximación descendente.* [Online] enero 30, 2004. [Citado: mayo 13, 2008.] Estructuración de la Información: Aproximación descendente.

15. **Hassan Montero, Yusef and Núnez Peña, Ana. 2005.** Diseño de Arquitecturas de Información: Descripción y Clasificación. *Diseño de Arquitecturas de Información: Descripción y Clasificación.* [Online] enero 14, 2005. [Citado: mayo 15, 2008.] http://www.nosolousabilidad.com/articulos/descripcion_y_clasificacion.htm.

16. **Hassan, Yusef. 2005.** Ejemplo de prueba con CardSword. *Ejemplo de prueba con CardSword.* [Online] marzo 17, 2005. [Citado: mayo 4, 2008.] http://www.bitacoras.sidar.org/g4/index.php?2005/03/17/7.

17. **Hernandez Valverde, Pedro. 2004-2005.** Una de las metodologías de gestión de proyectos más innovadoras. *Una de las metodologías de gestión de proyectos más innovadoras.* [Online] 2004-2005. [Citado: mayo 7, 2008.] http://is.ls.fi.upm.es/doctorado/Trabajos20042005/Hernandez.pdf.

18. La diagramación en la arquitectura de información. *La diagramación en la arquitectura de información.* [Online] diceimbre 25, 2007. [Citado: mayo 17, 2008.] http://nosolousabilidad.com/articulos/diagramacion.htm.

19. **León Pavón, Eduardo ;. 2 abril,2007.** Visual Paradigm, una herramienta de los mas util. *Visual Paradigm, una herramienta de los mas util.* [Online] abril 2, 2 abril,2007. [Citado: mayo 8, 2008.] http://slion2000.blogspot.com/2007/04/visual-paradigm-una-herramienta-de-lo.html.

20. **López Rodríguez, César. 2003.** Ejemplo de desarrollo software utilizando la metodología RUP . *Ejemplo de desarrollo software utilizando la metodología RUP .* [Online] julio 28 , 2003. [Citado: mayo 5, 2008.] http://www.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/.

21. **Maurer, D. and Warfel, T. 2004.** Card sorting: a definitive guide. En Boxes and Arrows. *Card sorting: a definitive guide. En Boxes and Arrows.* [Online] abril 2004. [Citado: mayo 17, 2008.] http://www.boxesandarrows.com/archives/card_sorting_a_definitive_guide.php.

22. **Miguel Angel. 2006.** Creando pruebas de Card Sorting (agrupación de tarjetas) con CardSword. *Creando pruebas de Card Sorting (agrupación de tarjetas) con CardSword.* [Online] abril 10, 2006. [Citado: mayo 4, 2008.] http://mi-blog.com/migue/2006/08/10/creando-pruebas-de-card-sorting-agrupacion-de-tarjetas-con-cardsword/.

23. Modelo de proceso de la ingenieria de la usabilidad y de la accesibilidad.MPIu+a. *Modelo de proceso de la ingenieria de la usabilidad y de la accesibilidad.MPIu+a.* [Online] julio 13, 2005. [Citado: mayo 3, 2008.] http://griho.udl.es/mpiua/mpiua/software.htm.

24. **Montero, Hassan and Yusef. 2006.** Indización Social y Recuperación de Información. *Indización Social y Recuperación de Información.* [Online] noviembre 16, 2006. [Citado: mayo 17, 2008.] http://www.nosolousabilidad.com/articulos/indizacion_social.htm.

25. **Nielsen, Jackob and Sano, Darrel. 2000.** Design of SunWeb - Sun Microsystems' Intranet. *Design of SunWeb - Sun Microsystems' Intranet.* [Online] 2000. [Citado: mayo 5, 2008.] http://www.useit.com/papers/sunweb/.

26. **NIELSEN, JACKOB. 2004.** Card Sorting: A cuántos usuarios se necesita evaluar . *Card Sorting: A cuántos usuarios se necesita evaluar .* [Online] agosto 12, 2004. [Citado: mayo 15, 2008.] http://www.proyectoweb.cubaweb.cu/boletin/card-sorting-a-cuantos-usuarios-se-necesita-evaluar.html.

27. **Ortega Santamaría, Sergio. 20005.** Desarrollo Conceptual y la técnica de Card Sorting. *Desarrollo Conceptual y la técnica de Card Sorting.* [Online] diciembre 14, 20005. [Citado: mayo 17, 2008.] http://www.nosolousabilidad.com/articulos/desarrollo_conceptual.htm.

28. **Pavon Mestras, Juan.** Rational Rose. *Rational Rose.* [Online] [Citado: mayo 5 , 2008.] http://www.fdi.ucm.es/profesor/jpavon/is2/Lab01RationalRose.pdf.

29. Rational Rose Enterprise Edition,. *Rational Rose Enterprise Edition,.* [Online] 2008. [Citado: mayo 5, 2008.] http://www.ciao.es/Rational_Rose_Enterprise_Edition__Opinion_612900.

30. Rational Rose. *Rational Rose.* [Online] [Citado: mayo 5, 2008.] http://www.slideshare.net/vivi_jocadi/rational-rose/.

31. **Rdickelman. 2006.** IBM EZSort: Card Sorting Software. *IBM EZSort: Card Sorting Software.* [Online] junio 24, 2006. [Citado: mayo 3, 2008.] http://www.epsscentral.info/knowledgebase/desdev/ibmezsort.

32. **Robertson, J. 2001.** Information design using card sorting. *Information design using card sorting.* [Online] 2001. [Citado: mayo 17, 2008.] http://www.steptwo.com.au/papers/cardsorting/.

33. **Ronda León, Rodrigo. 2005.** La Arquitectura de Información y las Ciencias de la Información. En No Solo Usabilidad. *La Arquitectura de Información y las Ciencias de la Información. En No Solo Usabilidad.* [Online] mayo 4, 2005. [Citado: mayo 17, 2008.] http://www.nosolousabilidad.com/articulos/ai_cc_informacion.htm.

34. **Rondan León, Rodrigo and Mesa Rábade, Yaima. 2005.** Análisis de Secuencia: una herramienta para la Arquitectura de Información. *Análisis de Secuencia: una herramienta para la Arquitectura de Información.* [Online] julio 6, 2005. [Citado: mayo 17, 2008.] http://www.nosolousabilidad.com/articulos/analisis_secuencia.htm.

35. **Rondan León, Rodrigo. 2008.** Arquitectura de Informacion:analisis histórico-conceptual. *Arquitectura de Informacion:analisis histórico-conceptual.* [Online] abril 28, 2008. [Citado: mayo 17, 2008.] http://nosolousabilidad.com/articulos/historia_arquitectura_informacion.htm.

36. **Schilb, Steffen.** About CardSort. *About CardSort.* [Online] [Citado: marzo 20, 2008.] http://www.cardsort.net/.

37. Sistemas de Clasificación de Información. *Sistemas de Clasificación de Información.* [Online] febrero 14, 2004. [Citado: mayo 15, 2008.] http://www.nosolousabilidad.com/articulos/sistemas_clasificacion.htm.

38. **Toro, Jorge A. 2008.** CardZort Zone . *CardZort Zone .* [Online] 2008. [Citado: mayo 3, 2008.] http://www.cardzort.com/cardzort/index.htm.

39. **Vizcaíno, Aurora ; García, Felix Oscar; l Caballero, Ismael.** Una Herramienta CASE para ADOO: Visual Paradigm. *Una Herramienta CASE para ADOO: Visual Paradigm.* [Online] [Citado: mayo 8, 2008.] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf.

40. Web Semántica: El papel del Arquitecto de la Información. *Web Semántica: El papel del Arquitecto de la Información.* [Online] abril 28, 2003. [Citado: mayo 15, 2008.] http://www.nosolousabilidad.com/articulos/web_semantica.htm.

ANEXOS

Preguntas de las Entrevistas

¿Sabe en qué consiste la Técnica del Card Sorting?

¿Se aplica en la UCI la Técnica? Cuáles son las causas que con llevan a esto?

¿Sabe si se ha hecho algún estudio o se ha desarrollado algo para mejorar el proceso?

¿Importancia que tiene el uso del Card Sorting en la UCI?

¿Que favorecería si se llegara a implementar un software que aplique el mecanismo?

GLOSARIO

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

Negocio: cualquier ambiente o entorno en cual está enmarcado el problema.

Patrón: solución común a un problema común de un determinado contexto.

Proceso del Negocio: funciones que se desarrollan en el ambiente o entorno que definimos como negocio.

Técnica: Método, táctica, procedimiento para hacer alguna cosa.