Universidad de las Ciencias Informáticas

Título: Gestionar el Componente del Rating Elo y su Integración a la Plataforma de Infodrez.

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor: Leonel Columbié Martínez

Tutor: MBA. Humberto González Hernández

Co-tutor: Msc. Isbel Herrera del Sol

Ciudad de la Habana_ Junio del 2008

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2008.

Leonel Columbié Martínez	MBA. Humberto González Hernández

AGRADECIMIENTOS

Agradezco a mis padres, abuelo y hermana por su amor e incondicionalidad.

A Isied por haber estado siempre a mi lado, por su amor.

A mi amigo y tutor Humberto por todo lo que he aprendido.

A mi consultante Alieski por toda su ayuda.

A mi tío Isnel por haber sido siempre fiel.

A Michel por su apoyo en momentos difíciles.

A Adrián, mi colega de módulo, por su esfuerzo.

A Adrián, alias la fiera por su invaluable ayuda.

A Rolando, alias la bestia por ser el amigo que ha sido.

A Dolennis por brindarme sus ideas.

A mi tío Bismark por tantos consejos.

A Félix por ser el colega de todo momento.

A mis compañeros y amigos, los de aquí y los de Sagua por ser y estar.

Especialmente agradezco a Néstor, quien ha sido un amigo y un padre.

A ti madre, Gracias.

DEDICATORIA

A mi madre…

Resumen

La Cátedra de Ajedrez en la Universidad de las Ciencias Informáticas (UCI) surgió como una necesidad de desarrollar el enorme potencial que desde sus inicios se hizo evidente que existía. Con la creación de la Cátedra Honorífica “Remberto Fernández”, y su posterior organización, se creó la necesidad de una Plataforma Integrada virtual que permitiera dar respuesta a las múltiples funciones que debían cumplirse para fomentar y consolidar la práctica del Ajedrez en la UCI. Una de las aristas fundamentales de la actividad ajedrecística es la deportiva, dentro de la cual la función del Arbitraje resulta básica para organizar los Torneos, asegurar el cumplimiento del Reglamento durante las partidas, registrar los resultados y computar todos los indicadores, incluido el Rating ELO. El Subsistema del Cálculo de la Variación del Rating Elo desarrolla el análisis, diseño e implementación de una Aplicación Web que permita obtener la Variación del Rating Elo de cada jugador una vez terminado un torneo, además de reportes necesarios al área solicitante.. El trabajo está estructurado en cuatro capítulos para la mejor comprensión del proceso de Ingeniería de Software llevado a cabo. Como resultado del mismo se implementaron las opciones relacionadas con el Cálculo de la Variación del Rating Elo, integrado en el Módulo de Arbitraje de Infodrez, considerándose preliminarmente que posee una capacidad operacional que le permite responder a los requerimientos formulados por el cliente.

PALABRAS CLAVE: Rating Elo, Variación, Reportes.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Introducción

Uno de los juegos de mayor rango en la formación de valores éticos y desarrollador del razonamiento lógico es, a no dudarlo, el Ajedrez. Sus orígenes se remontan muchos siglos atrás, y existen conjeturas diversas sobre la ubicación de sus comienzos, la más aceptada de las cuales lo establece en la India. En esencia, se trata de la porfía entre dos contendientes, cada uno de los cuales tiene un conjunto de piezas a su disposición (blancas uno, negras el otro) con movimientos preestablecidos, que deben mover simulando una batalla sobre un tablero de 64 casillas (8 x 8), con escaques de colores oscuros y claros, en turnos alternados, cumpliendo reglas registradas en un Reglamento, con el propósito de determinar un ganador [1].

Existe un reconocimiento universal acerca de que el aprendizaje del Ajedrez puede ser útil como forma de desarrollar el razonamiento. El Ajedrez es jugado tanto recreativa como competitivamente: en sesiones entre amigos, entre miembros de clubes, en torneos, en Internet, e incluso por correo (Ajedrez por correspondencia).

Cuba se hizo notar a nivel internacional en la práctica de este deporte, con la figura de José Raúl Capablanca y Graupera, quien fuera el tercer campeón mundial, entre 1921 y 1927. Luego, con el triunfo de la Revolución, se creó un movimiento verdaderamente masivo de la práctica del Ajedrez en nuestro país, liderado por el Comandante Ernesto Guevara, quien a pesar de sus múltiples obligaciones, encontró tiempo para mantenerse vinculado al juego ciencia, y vaticinó que algún día en Cuba tendríamos muchos Grandes Maestros, lo que se ha cumplido con creces, pues ya suman más de veinte (20) los titulados cubanos con el máximo pergamino de la FIDE. Muy recientemente, el GM Leinier Domínguez ha logrado superar la mítica cifra de los 2700 puntos de Rating ELO, lo que constituye uno de los resultados más relevantes del Ajedrez en Cuba en todos los tiempos [2].

En el primer Claustro de la UCI, el Comandante en Jefe Fidel Castro dedicó una de sus intervenciones a referirse a las bondades del aprendizaje, estudio y práctica del Ajedrez, en la formación del conocimiento y los valores de los estudiantes universitarios, y particularmente de los Ingenieros Informáticos. Entre las ideas esenciales de aquella jornada, destaca la siguiente:

"Hay que enseñar ajedrez en todo el país a partir del elemental análisis de que es una ciencia, la ciencia de las variantes y de la selección de las variantes y la informática es

el deporte que ha servido para demostrar que una computadora le puede ganar a un campeón."

Como resultado de este Claustro, y a partir de otros antecedentes del Ajedrez en la UCI, tomó fuerza la idea de fundar en el año 2003, la Cátedra Honorífica de Ajedrez "Remberto Fernández". Entre sus primeras acciones, se firmó un Convenio de Colaboración con el Instituto Superior Latinoamericano de Ajedrez (ISLA), refrendado por los Rectores de ambos Centros, que ha permitido desde entonces contar con la Asesoría de esta prestigiosa entidad. Como parte de esta relación de trabajo, el ISLA ha aportado la asesoría con profesionales del Ajedrez que han ofrecido conferencias y simultáneas en múltiples ocasiones, y ha asegurado la participación de jugadores de la UCI en los más importantes eventos que se celebran en nuestro país. La UCI por su parte, estableció una línea de trabajo para el desarrollo de aplicaciones informáticas que sirvan como soporte al programa nacional de enseñanza del Ajedrez en los diferentes niveles de educación, sitios Web de eventos nacionales e internacionales celebrados en Cuba que incluyan: partidas, resultados, lugares, mini biografías de los participantes, etc; el más importante de los cuales es el Proyecto Infodrez, al que pertenece el módulo a que se refiere el presente Trabajo.

Considerando la importancia del arbitraje en la práctica del Ajedrez de Alto Rendimiento, a partir de la necesidad de formalizar los Emparejamientos y el Calculo del Rating ELO, se decidió que el módulo de Arbitraje se incorporara entre los que serían concebidos y desarrollados en la primera etapa del Proyecto.

Se impuso entonces realizar una investigación sobre las aplicaciones existentes que abordaran el Cálculo del Rating ELO, determinándose que las que existen (Swiss Manager y Swiss Perfect las más populares) presentan un bajo nivel de Integración con otras aplicaciones relacionadas con aspectos diversos que están concebidos en nuestra Plataforma. Adicionalmente, no se encontró ningún sistema informático en plataformas de Software Libre, que aborde integralmente el tema del Arbitraje.

Tomando en consideración los elementos planteados en el párrafo precedente, y el hecho de que en la UCI fue necesario suspender la gestión y publicación del Rating ELO UCI, por la imposibilidad de llevar manualmente este complejo sistema de puntuación, se considera que el módulo de Arbitraje de Infodrez debe constituir una valiosa herramienta para la Cátedra de Ajedrez de nuestro Centro, y para el personal que labora en cualquier lugar donde se juegue ajedrez de manera oficial. Considerando lo anterior se presenta la siguiente situación problémica:

En nuestra Universidad no existe en este momento una herramienta idónea para procesar la gran cantidad de información del cálculo del Rating ELO para cada uno de los ajedrecistas, que se genera luego de la culminación de los torneos, por lo que se hace necesario desarrollar una aplicación informática para la gestión, entre otras, de la siguiente información: Rating Elo inicial provisional (para Jugadores que no poseen Rating), Expectancia, y Rating ELO final. Se necesita además actuar con inmediatez, sólo basta preguntarse si con la información que se cuenta en estos momentos se puede dar respuesta por ejemplo, a las siguientes preguntas:

- ¿Se podrá ordenar el Rating ELO de los jugadores de la UCI, permitiendo conformar y publicar un Escalafón único?
- ¿Cual fue el jugador que más incrementó su Rating ELO en cada uno de los Torneos que se desarrollen en la UCI?
- ¿Podrá generalizarse esta experiencia a otras Federaciones Locales en el país, de manera que pueda considerarse la posibilidad de establecer sistemas de Rating ELO: Provinciales, Municipales, para Categorías Escolares, y otras ?

Resulta evidente que se requiere información actualizada y confiable para conocer las tendencias de los torneos en la UCI o cualquier parte donde estos puedan celebrarse, basados en datos actualizados de manera constante. Dando respuesta a estos problemas surge la Plataforma Infodrez, y como componente de ésta, el Sistema o Aplicación Web para el Cálculo del Rating ELO.

Con este sistema se obtendrá un método rápido, eficaz y confiable de controlar el Rating ELO y sus variaciones al concluir cada torneo. El Sistema será otra de las herramientas que se utilicen en la Universidad para gestionar los procesos de manera automatizada; lo cual permitirá un mayor control, de forma más confiable, con un gasto mínimo de recursos. Consideramos que el principal aporte será que la Cátedra de Ajedrez podrá contar en tiempo real con la información de sus Torneos, y cada uno de los Jugadores tendrá acceso a su Rating ELO actualizado, el que estará variando dinámicamente con los resultados que logre en cada uno de los torneos en los que participe, lo que de seguro incentivará el incremento de la competitividad y la práctica del Ajedrez en la UCI.

Se hace necesario entonces, precisar el siguiente **problema científico:**

¿** Cómo gestionar el Cálculo del Rating Elo en la Universidad de las Ciencias Informáticas **?

El **objeto de estudio** del mismo es**:** *"Gestión del Cálculo del Rating ELO y obtención de reportes que procesen la información que se genera en los torneos"* y su **campo de acción** será**:** *Arbitraje en el Ajedrez*; aunque debemos señalar que en etapas posteriores se prevé generalizar su alcance fuera de la Universidad, e inclusive, existe una visión internacional del Proyecto.

Teniendo en cuenta lo anterior, se plantea la siguiente: **Idea a defender**:

El desarrollo de una Aplicación Web que gestione el Cálculo del Rating Elo, permitirá una mayor eficiencia de los procesos de arbitraje, en los torneos de ajedrez en la UCI.

Determinándose como **objetivo general** de la Investigación, el siguiente:

Gestionar el componente del Rating Elo y su Integración a la Plataforma Infodrez.

Se derivan de éste, los siguientes **objetivos específicos**:

- ✓ Estudiar los procesos del negocio actual.
- ✓ Realizar análisis y diseño del sistema propuesto.
- ✓ Implementar el resultado del análisis y el diseño, considerando los torneos en la modalidad Round Robin.

Para darle cumplimiento a esos objetivos se ejecutarán las siguientes **tareas**:

- ➢ Realizar un estudio del entorno de trabajo.
- ➢ Identificar las necesidades del cliente.
- ➢ Declarar los requisitos que debe cumplir el sistema.
- ➢ Describir los procesos que se van a implementar en el sistema.
- ➢ Especificar los procesos que se van a implementar en el primer ciclo de desarrollo.
- ➢ Modelar conceptualmente las clases que están implicadas en el sistema.
- ➢ Desarrollar los diagramas de actividad.
- ➢ Desarrollar los diagramas que describen el diseño del sistema.
- ➢ Describir las clases del diseño.
- ➢ Implementar la aplicación

El presente trabajo está compuesto por cuatro capítulos; los cuales servirán de guía en la comprensión del proceso llevado a cabo para dar cumplimiento a los objetivos trazados, los mismos describen paso a paso todo el contenido necesario de investigación con resultados objetivos y eficientes para la gestión del Cálculo de la Variación del Rating Elo en la Universidad de las Ciencias Informáticas.

A continuación se muestra una breve descripción de cada Capítulo.

Capítulo I. Fundamentación Teórica:

Se hace un estudio del estado del arte, así como un análisis del por qué de las herramientas, tecnologías y lenguajes utilizados en el desarrollo e implementación del sistema propuesto, además de la fundamentación matemática del Cálculo del ELO.

Capítulo II. Características del sistema:

Se hace un análisis del proceso de negocio existente y a partir del mismo se obtienen las actividades que serán sujetas a automatización. Por otro lado se enuncian los requerimientos funcionales y no funcionales. Se hace, además, una descripción de los casos de uso que serán implementados, y se muestra el diagrama de casos de uso del sistema.

Capítulo III. Análisis y diseño del sistema:

Se obtiene, a través de los casos de uso del sistema, el diagrama de análisis. Como resultado del análisis obtenemos el diseño de la aplicación, el cual queda expresado en el diagrama de clases del diseño, los diagramas de secuencia y la descripción de las clases. Además en este Capítulo se hace el análisis de costo beneficio de la aplicación.

Capítulo IV Implementación:

Se describe la distribución del sistema propuesto a través del diagrama de despliegue. También se muestra el diagrama de componentes, el cual ayuda a la comprensión del flujo de trabajo de implementación.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1. Introducción

La utilización de Aplicaciones para el Cálculo del Rating Elo es la evolución lógica de cualquier sistema informático que produzca información en sus procesos de negocio. La Variación del Rating Elo lo utiliza la Federación Internacional de Ajedrez (FIDE) para valorar la calidad de los ajedrecistas, establecer normas para recibir y confirmar los títulos, clasificar los torneos, etc. En 1970 la FIDE aprobó el sistema de coeficientes individuales elaborado por el Doctor norteamericano Arpad E. Elo, profesor de Física de la Universidad Marquette de Wilwaukee, Wis; sistema que ha sido internacionalmente reconocido para medir el rendimiento de los Jugadores. En 1971 se comenzó a publicar la Lista internacional del Rating ELO [3].

El sistema Elo se basa en dos principios fundamentales:

- ✓ Un coeficiente dado, cuya cifra guarda relación con las actuaciones de los jugadores.
- ✓ El principio de que la posibilidad de triunfo entre dos adversarios guarde relación con los coeficientes (rating) que ambos posean.

La formula del Elo es la siguiente:

Rn = Ro + K (W – We)

Leyenda:

Rn = Rating Nuevo

Ro = Rating viejo

K = Constante

W = Puntos alcanzados

We = Expectancia

Los valores del coeficiente K pueden ser:

K 25 = Menos de 30 partidas o 3 torneos y un Elo inferior a 2300.

K 15 = Más de 30 partidas o 3 torneos y un Elo inferior a 2400.

K10 = Cuando alguna vez se obtuvo un Elo superior a 2400.

1.1 Estado del Arte

Este acápite se ha organizado diferenciando el estudio de las aplicaciones informáticas que abordan el cálculo del Rating ELO: para el escenario internacional, para Cuba, y específicamente para la UCI [12].

1.1.1 En el mundo

En la actualidad existen en el mundo muchas aplicaciones que abordan este problema, de donde surgen las preguntas: ¿Tenemos acceso a ellas, satisfacen nuestras expectativas ?, dando respuesta negativa en ambos casos con alto índice de seguridad, lo que a su vez nos aporta las razones para una segunda pregunta: ¿Cómo podemos desarrollar algunas de estas soluciones a partir del potencial con el que contamos?. En el ámbito internacional existen muchos sitios que publican contenidos relacionados con los procesos arbitrales, dichos sitios son de gran valor dado que publican la información necesaria para que los árbitros se mantengan actualizados; algunos ejemplos de estos sitios son el sitio oficial de la FIDE (www.FIDE.com), el de la Federación Española de Ajedrez (www.FEDA.org), estos sitios aún tienen algunas carencias en los temas relacionados con el arbitraje. El Portal que lleva la delantera en los temas específicos del arbitraje de ajedrez en idioma español es: www.arbitrosdeajedrez.com, que a pesar de estar ahora en una etapa de mantenimiento, es el que más se centra en los temas específicos del arbitraje. Estos Sitios se enfocan fundamentalmente al tema de los contenidos, no así a la gestión.

Otros productos, Swiss-Manager y Swiss-Perfect los más destacables, se enfocan a la gestión arbitral en los Torneos; pero presentan las importantes limitaciones de ser software propietario y de poseer una muy débil integración con otros productos del sector, lo que obliga a registrar la misma información en varios soportes diferentes, sin posibilidad alguna de reutilización de contenidos [2].

1.1.2 En Cuba

No existe una generalizada cultura de trabajo con aplicaciones informáticas que gestionen el cálculo del Rating ELO en nuestro país, y sólo en los Torneos formales de algún nivel se recurre a ellas, tratándose en todos los casos de las mencionadas anteriormente, o de alguna otra con menor rigor profesional. En todos los casos se trata de aplicaciones de escritorio que imponen restricciones importantes de conectividad, lo que imposibilita concebir una Base de Datos Nacional de Jugadores, e impide, de facto, concebir una gestión eficaz del Rating Elo Nacional.

A nivel nacional lo que más se acerca al paradigma que estamos concibiendo es el Sitio de ajedrez en Cuba (www.inder.cu/capablanca), en el que al igual que los Sitios de las federaciones internacionales, se trata el tema del arbitraje muy superficialmente y los árbitros en la mayoría de los casos tienen que recurrir a los sitios externos [13].

1.1.3 En la UCI

La situación que presenta la UCI fue abordada con anterioridad en este Trabajo. El análisis que se derivó de la misma, llevó a la decisión de comenzar el desarrollo de una aplicación que cubra las expectativas planteadas. Este Sitio Web permitirá la actualización permanente del Rating Elo de cada jugador de manera dinámica, brindando la información necesaria y exacta para cada ajedrecista en la UCI. En nuestra universidad ya contamos con el portal Infodrez, en el cual se tratan bastante a fondo los temas relacionados con el Ajedrez; pero aún existe carencia de una sesión especializada en los temas del arbitraje porque aunque el portal cuenta con una sesión dedicada al mismo, esta solo se encarga de publicar noticias relacionadas con el tema, lo que implica no contar con una aplicación capaz de calcular la variación del Rating Elo, al terminar un torneo .

1.1.3.1 ¿ Por qué nuestra propuesta ?

La razón fundamental de nuestra propuesta radica en dotar a los árbitros de una herramienta que les permita automatizar todos lo procesos posibles, siendo esta herramienta software libre, y soportada sobre una plataforma Web integrada (Infodrez) que contará con otros temas relacionados con el ajedrez que le resultarán muy útiles a los árbitros, en este caso específicamente para el cálculo de la variación del Rating Elo.

1.2 Metodologías

Para desarrollar cualquier software es necesario guiar el proceso a través de una metodología, la cual será la encargada de elaborar “el plano” sobre el cual se apoyará el equipo de desarrollo. En la actualidad existen diferentes metodologías por las que se guía el Proceso Unificado de Desarrollo de Software, entre ellas se encuentran: MSF, RUP y xP como las más utilizadas. Estas fueron estudiadas con el fin de escoger una metodología para nuestro desarrollo. A continuación se describen los elementos esenciales de cada una de ellas.

1.2.1 MSF (Microsoft Solution Framework)

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en segundo plano las elecciones tecnológicas.

Originalmente creado en 1994 para conseguir resolver los problemas a los que se enfrentaban las empresas en sus respectivos proyectos, se ha convertido posteriormente en un modelo práctico.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

Figura 1.1 Metodología MSF

1.2.2 xP (Extreme Programming)

¿**Que es xP**?:

Metodología para un ágil desarrollo de software.

Programación basada en los deseos del cliente.

El equipo lo conforman los jefes de proyecto, desarrolladores y el cliente.

Se rige por valores y principios.

Dentro de sus **Valores** fundamentales se encuentran:

Comunicación: Crear software requiere de sistemas comunicados.

Simplicidad: Empezar con lo necesario y requerido y trabajar desde ahí.

Retroalimentación: Del sistema, del cliente, y del equipo.

Sus **Actividades** aseguran evitar coeficientes de errores que superen la media permitida, citando algunas como:

Codificación: La parte más importante de xP.

Pruebas: Nunca se puede estar seguro de algo hasta haberlo probado.

Diseño: Crear una estructura del diseño para evitar problemas.

A continuación se muestra el **Ciclo** de trabajo que se concibe:

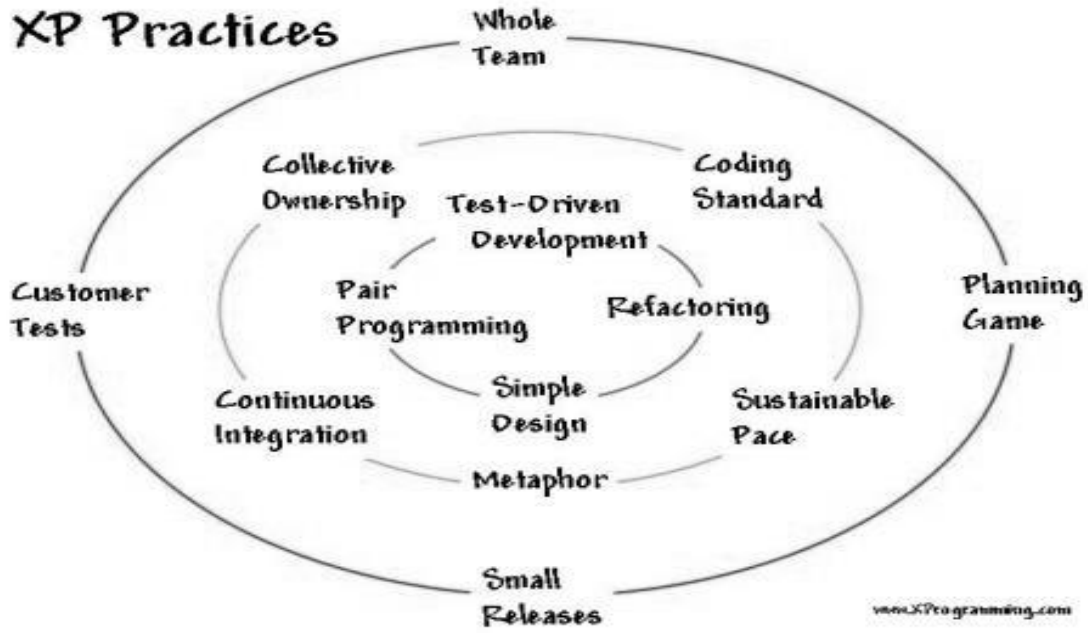


Figura 1.2 Ciclo de la Metodología xP

Esta metodología se basa en ir construyendo el software y hacerle pruebas al mismo tiempo de modo que el cliente -que se convierte en parte del equipo-; que permite construir el software de la manera que el cliente quiera. La metodología está diseñada para entregar al cliente el software que necesita, cuando lo necesita. Extreme Programming se usa actualmente para la creación y desarrollo práctico de software.

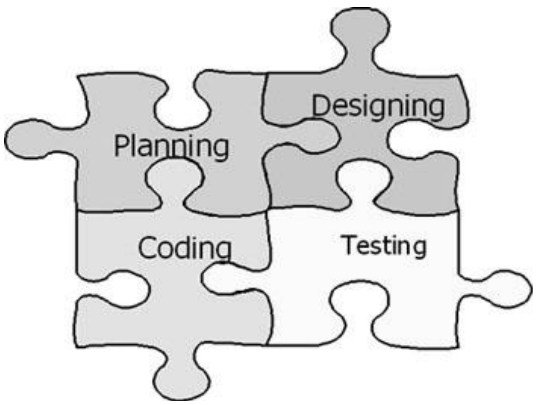


Figura 1.3 Metodología XP.

Algunas de las ventajas que tiene la metodología xP son:

I. El cliente tiene el control sobre las prioridades.

II. Se hacen pruebas continuas durante el proyecto.

III. La xP es mejor utilizada en la implementación de nuevas tecnologías donde los requerimientos cambian rápidamente

IV. Programación organizada.

V. Menor taza de errores.

VI. Satisfacción del programador.

VII. Integración continua y el monitoreo de las métricas del código apoyan a la XP para conseguir resultados rápidos con software muy seguro, resistente a "refactoring" y fácil de mantener.

La metodología xP o Extreme Programming es una de las variantes de las metodologías ágiles con más aceptación en la comunidad internacional de desarrollo. Su creador, Kent Beck la comenzó a gestar junto con Ward Cunningham en 1990 y tomó su forma final en 1996. Los fundamentos de la xP según Beck son: mejorar la comunicación, buscar la simplicidad, buscar retroalimentación en que tan bien va nuestro trabajo y siempre hay que proceder con valentía. Una de las herramientas más importantes de la xP es el desarrollo orientado a pruebas, que utiliza las pruebas unitarias como eje de todo desarrollo. Siendo una directriz de esta herramienta no escribir código para el que no tengamos previamente una prueba unitaria, no sólo mejoramos la seguridad del desarrollo, sino que también documentamos el código de producción con el código de pruebas.

1.2.3 RUP (Rational Unified Process)

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

Inicio: permite determinar la visión del proyecto.

Elaboración: se determina la arquitectura base.

Construcción: se propone obtener la capacidad operacional inicial.

Transición: debe llegarse a obtener la versión entregable del proyecto.

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, superponiendo unas etapas con otras. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

RUP está compuesto por tres elementos fundamentales

Actividades: son los procesos que se llegan a determinar en cada iteración.

Trabajadores: son las personas o entes involucrados en cada proceso.

Artefactos: un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Dentro de cada iteración de cada fase se llevan a cabo nueve flujos de trabajo dentro de los cuales los seis primeros son llamados de ingeniería y los demás son de apoyo.

- **Flujos de Trabajo de Ingeniería**

Modelado del negocio: Este flujo identifica los procesos de negocio, los que estarán sujetos a automatización y quiénes intervienen en los mismos.

Requerimientos: Se identifican las restricciones que se imponen y lo que el sistema debe hacer.

Análisis y Diseño: Describe cómo el programa será realizado y define cómo será programado.

Implementación: Define cómo estarán los nodos ubicados y la ubicación de los objetos y clases en paquetes.

Prueba: Se localizan los defectos del software.

Instalación: Se entrega una versión operacional.

- **Flujos de Trabajo de Apoyo**

Administración de proyecto: Encargado de organizar el trabajo y de que se termine el proyecto en el tiempo previsto.

Administración de configuración y cambio: Describe el uso y actualización concurrente de los elementos, control de versiones entre otras actividades.

Ambiente: Describe los procesos y herramientas que soportarán al equipo de trabajo del proyecto.

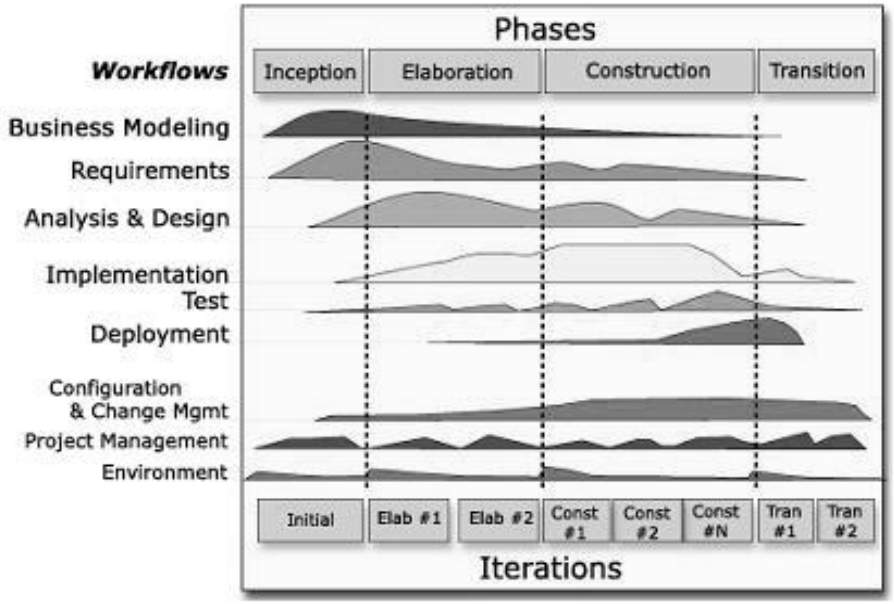


Figura 1.4 Fases e iteraciones de la Metodología RUP

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. Esta metodología es muy utilizada en el proceso de desarrollo de software por ser flexible; además, al ser iterativa, permite que se vaya construyendo el software por ciclos, por lo cual se pueden detectar errores con tiempo de antelación. Es una metodología confiable pues desde su surgimiento ha tenido una gran aceptación [4][10].

Por todos estos elementos el sistema propuesto estará guiado por la metodología del Proceso Unificado de Desarrollo de Software quien además es recomendada para proyectos grandes, es de decir de gran alcance como lo es el caso de Infodrez.

1.3 Lenguajes a utilizar

1.3.1 UML

El Lenguaje Unificado de Modelado (UML - Unified Modeling Language) ayuda a especificar, visualizar, y documentar modelos de sistemas de software, incluyendo su estructura y diseño. La actual versión de UML (UML 2.0) cuenta con treinta tipos de diagramas standard. UML fue originalmente creado por Rational Software, pero actualmente es atendido por el Object Management Group (OMG). Es desde finales de la década del 90, un lenguaje de modelado orientado a objetos estándar, de acuerdo con el Object Management Group (OMG). UML es un estándar de la industria, pero no sólo de la industria del software sino, en general, de cualquier industria que requiera la

construcción de modelos como condición previa para el diseño y posterior construcción de prototipos.

Además ayuda a entender la realidad de la tecnología y el funcionamiento de los procesos de negocio, reduciendo el costo y el tiempo empleado en la construcción de las piezas que constituirán el modelo. Este lenguaje reúne una serie de propiedades como:

- ✓ Es ampliamente utilizado por la industria desde su adopción por OMG.
- ✓ Modela estructuras complejas.
- ✓ Las estructuras más importantes que soporta tienen su fundamento en las tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos.
- ✓ Modela el comportamiento del sistema mediante casos de uso, diagramas de secuencia y de colaboración, que sirven para evaluar el estado de este [8].

1.3.2 PHP

PHP (acrónimo recursivo de "PHP: Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. PHP puede ser utilizado en cualquiera de los principales sistemas operativos, incluyendo Linux, en variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS; o sea es multiplataforma. PHP es soportado por la mayoría de los servidores Web de hoy en día, por ejemplo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro Server, Caudium, Xitami, OmniHTTPd y muchos otros. De modo que con PHP se puede elegir el sistema operativo a utilizar y el servidor que se prefiera.

Otro de los aspectos a destacar es que al ser PHP un lenguaje que se ejecuta en el servidor, no es necesario que un navegador en particular lo soporte, es independiente del navegador, sin embargo, para que sus páginas funcionen, el servidor donde están alojadas debe soportar PHP. Además PHP cuenta con una gran comunidad de desarrolladores, como producto de código abierto, gozando de la ayuda de un gran grupo de programadores, lo que permite que los fallos de funcionamiento se encuentren y reparen rápidamente.

Otro de los aspectos que se tuvieron en cuenta fue que PHP está dentro de los lineamientos trazados por la Dirección de Informatización de la UCI en cuanto al uso de lenguajes de programación para el desarrollo de aplicaciones Web.

Entre otras ventajas tenemos:

- ✓ Muy sencillo de aprender.
- ✓ Similar en sintaxis a C y a PERL
- ✓ Soporta en cierta medida la orientación a objetos. Clases y herencia.
- ✓ El análisis léxico para recoger las variables que se pasan en la dirección lo hace PHP de forma automática. Librándose el usuario de tener que separar las variables y sus valores.
- ✓ Se puede incrustar código PHP con etiquetas HTML.
- ✓ Excelente soporte de acceso a base de datos.
- ✓ Se puede hacer de todo lo que se pueda transmitir por vía HTTP.
- ✓ PHP no soporta directamente punteros, como el C, de forma que no existen los problemas de depuración provocados por estos.
- ✓ Se pueden hacer grandes cosas con pocas líneas de código.
- ✓ Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor (acceso a base de datos, encriptación, envió de correo, gestión de un e-commerce, xml, creación de PDF ...)
- ✓ Esta siendo utilizado con éxito en varios millones de sitios web.
- ✓ Es multiplataforma, funciona en todas las plataformas que soporten apache.
    - ✓ Es software libre. Se puede obtener en la web y su código esta disponible bajo la licencia GPL.

1.3.2.1 PHP 5

En la implementación del sistema propuesto será utilizado PHP5. Esta es la versión más actual de este lenguaje. La selección se basa en que en esta versión, entre otras cosas, hace un cambio en el manejo de los objetos. Por ejemplo en PHP4 los objetos son tratados igual que otros tipos de datos básicos, como los enteros o los arreglos.

O sea, cuando se realizan operaciones sobre los objetos, como asignación de variables o cuando son pasados como parámetros a funciones, todo el objeto es copiado [7].

Por otra parte, en PHP5 todas las variables que nombran objetos son en realidad referencias. Otro de los elementos es que en PHP5 se incluyen modificadores de control de acceso para implementar el encapsulamiento, lo cual no existía en versiones anteriores. PHP tiene Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.

Por todo lo anterior se decidió utilizar a PHP como el lenguaje encargado de modelar el sistema que se propone [4].

1.4 Sistema Gestores de Base de Datos (SGBD)

Un Sistema Gestor o Manejador de Bases de Datos (SGBD) es un conjunto de programas que permite a los usuarios crear y mantener una BD, por lo tanto, el SGBD es un software que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones. Pueden ser de propósito general o específico. Actualmente existen varios SGBD entre ellos: Oracle, Visual Fox Pro, PostgreSQL, Microsoft SQL Server 2000, MySQL, el cual será utilizado en la implementación del sistema propuesto.

1.4.1 Microsoft SQL Server 2000

Microsoft SQL Server 2000 es un sistema cliente-servidor de administración de base de datos relacional (Relational Database Management System - RDBMS) diseñado para procesamiento de transacciones en línea de alto rendimiento (online transaction processing - OLTP), almacenamiento de datos (data warehousing) y aplicaciones de comercio electrónico (e-commerce). Proporciona seguridad, fiabilidad y escalabilidad para poner en marcha cualquier aplicación en un tiempo pequeño, destacando sus sencillas tareas de administración y su capacidad de analizar la información. Tratándose de un Gestor de Bases de Datos propietario, se descarta su utilización.

1.4.2 PostgreSQL

PostgreSQL es un sistema gestor de base de datos con características a destacar a nivel mundial, por sus funcionalidades se hizo un estudio detallado del mismo encontrando resultados satisfactorios los cuales mostraremos a continuación para una mejor comparación entre PostgreSQL y el seleccionado por el proyecto INFODREZ.

Dentro de sus características tenemos:

- ✓ PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son

consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transactions, optimización de consultas, herencia, y arrays.

- ✓ PostgreSQL soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario.
- ✓ PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.
- ✓ PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- ✓ PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.
- ✓ La característica de PostgreSQL conocida como *Write Ahead Logging* incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos. Esto puede ser enormemente beneficioso en el caso de caída, ya que cualesquiera cambios que no fueron escritos en la base de datos pueden ser recuperados usando el dato que fue previamente registrado. Una vez el sistema ha quedado restaurado, un usuario puede continuar trabajando desde el punto en que lo dejó cuando cayó la base de datos.
- ✓ PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.
    - ✓ La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

1.4.3 MySQL

MySQL es un sistema de gestión de bases de datos relacionales. Una base de datos relacional almacena datos en tablas separadas en lugar de poner todos los datos en

un gran almacén, lo que añade velocidad y flexibilidad. La parte SQL de "MySQL" se refiere a "Structured Query Language".

El SQL es el lenguaje estandarizado más común para acceder a bases de datos. MySQL es Open Source, lo que quiere decir que es posible para cualquiera usar y modificar el software. Cualquiera puede bajar el software MySQL desde Internet y usarlo sin pagar nada. Si lo desea, puede estudiar el código fuente y cambiarlo para adaptarlo a sus necesidades.

El servidor de base de datos MySQL es muy rápido, fiable y fácil de usar. MySQL Server trabaja en entornos cliente/servidor o incrustados. Además, funciona en diferentes plataformas y fue escrito en C y en C++. Este gestor de base de base de datos será el utilizado por sus innegables ventajas expuestas anteriormente, pero también porque la mayoría de las bases de datos con que se cuenta en estos momentos en la universidad corren sobre él. Otro aspecto que influyó en su elección es la experiencia con que se cuenta en la Universidad de las Ciencias Informáticas en el uso de este gestor, lo cual constituye sin duda una poderosa razón pues la base de datos que soportará, podría ser atendida por personas ajenas a la elaboración del software, sin necesidad de un estudio previo de un gestor desconocido; facilitando su mantenimiento.

1.4.4 Servidor de aplicaciones Apache.

Apache es un servidor HTTP de código abierto el cual soporta tanto sistemas operativos UNIX como Windows NT. Es un servidor seguro y eficiente que cumple con los actuales standards de HTTP. Otra de sus características es que contiene mensajes de errores altamente configurables. Apache es utilizado como servidor HTTP en el 56.00 % de los sitios publicados en Internet. Fue creado en 1995 como parche a un servidor de aplicaciones llamado NCSA HTTP. Apache es un servidor altamente configurable de diseño modular. Actualmente existen muchos módulos para Apache que son adaptables a este, los cuales se clasifican en tres categorías:

- **Módulos Base**: Módulo con las funciones básicas del Apache.
- **Módulos Multiproceso**: Son los responsables de la unión con los puertos de la máquina, acepando las peticiones y enviando a los hijos a atender a las peticiones
- **Módulos Adicionales**: Cualquier otro módulo que le añada una funcionalidad al servidor.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código. El resto de funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software.

1.5 Herramientas Utilizadas

1.5.1 Rational Rose

Existen herramientas Case de trabajo visuales como Analise, Designe, Rational Rose, que permiten realizar el modelado del desarrollo de los proyectos. En la actualidad una de las más utilizadas en el mercado es Rational Rose y es la que se utiliza en la modelación de este proyecto. Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. El navegador UML de Rational Rose nos permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Incluye un conjunto de herramientas de Ingeniería Inversa y generación de código que facilitan el producto hasta el producto final. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción.

1.5.2 Visual Paradigm

Es una herramienta de modelado diseñada para un gran número de usuarios, incluyendo ingenieros de sistemas, analistas de sistemas, analistas de negocio, arquitectos de sistemas. Además se integra con IDEs (Eclipse, JBuilder, NetBeans, IntelliJ IDEA, JDeveloper and WebLogic Workshop) para soportar la fase de implementación de desarrollo de software. La transición desde el análisis al diseño y después a la implementación es cuidadosamente integrada dentro de la herramienta CASE, de esta manera se reduce significativamente el esfuerzo en todas las etapas del ciclo de vida del desarrollo del software. Incluye además un conjunto de herramientas que soportan Object-Relational Mapping (ORM), como Hibernate, lo cual

incluye generación completamente orientada a objetos, listo para usar librerías para obtener y modificar registros de base de datos para una gran variedad de gestores de base de datos, y la sincronización entre los diagramas de clases y los diagramas de entidad-relación (ERD). Presenta además generación de código e ingeniería inversa del código. Permite generar los EJB y así mismo los descriptores de despliegue para varios servidores de aplicación. A diferencia de muchas herramientas de modelado, pueden extenderse sus diagramas hechos desde el Visio y de Rational Rose.

Visual Paradigm Smart Development Environment (SDE) Enterprise Edition: Visual Paradigm SDE ofrece integración de la herramienta de modelado UML con los IDEs (Visual Studio®, Eclipse/WebSphere®, Borland JBuilder®, NetBeans/Sun™ ONE, IntelliJ IDEA™, Oracle JDeveloper, BEA WebLogic Workshop™). Visual Paradigm SDE se incrusta él mismo dentro de tu IDE favorito para proveer ambiente de desarrollo y modelado unificado. Dando todas las prestaciones y servicios que brinda el Visual Paradigm for UML Enterprise Edition. Visual Paradigm DB Visual Architect: DB Visual Architect es un simple y fácil de usar ambiente de Object-Relational Mapping (ORM), como Hibernate, el cual actúa como un puente entre el modelo de objetos, el modelo de datos y el modelo relacional. Este ayuda a los desarrolladores a construir aplicaciones de base de datos de alta calidad, mucho más rápido, mejores y barato. Disminuye el costo de desarrollo a un 75 % por automatizar el proceso de mapeo entre los objetos de java y las tablas de la base de datos visualmente y a través de diagramas. Los desarrolladores ahorran enormes cantidades de tiempo usando DB Visual Architect para manipular el código de SQL y JDBC transparentemente con (POJO), como DB Visual Architect está habilitado con ingeniería inversa de una base de datos relacional a un diagrama de clases y un diagrama entidad-relación (ERD) y viceversa (crear una base de datos relacional de modelos de relación de entidad (ERD) o de modelos de clases [7].

1.5.3 Macromedia Dreamweaver 8

Dreamweaver es un editor WYSIWYG que es el acrónimo de What You See Is What You Get (en inglés, "lo que ves es lo que obtienes"). Será usado para el diseño Web de la aplicación por las facilidades que brinda. Esta herramienta brinda soporte para aplicaciones PHP y además facilita el uso de las CSS acrónimo Cascade Style Sheet (en ingles, “hoja de estilo en cascada”).

1.5.4 Zend Development Environment

Este IDE, acrónimo de Integrated Development Environment (en ingles, "entorno integrado de desarrollo") es una de las herramientas más potente de programación para el lenguaje PHP. Cuenta con analizadores de código, permite completamiento de código entre otras ventajas. El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. Lo más destacable es que contiene una ayuda contextual con todas las librerías de funciones del lenguaje que asiste en todo momento ofreciendo nombres de las funciones y parámetros que deben recibir.

1.6 Arquitectura

Los tópicos más importantes en la arquitectura de software son ADLs, estilos y patrones. Los estilos expresan la arquitectura en el sentido más formal y teórico, son un tratamiento estructural que concierne más bien a la teoría, la investigación académica y la arquitectura en el nivel de abstracción más elevado, mientras que los patrones se ocupan de cuestiones que están más cerca del diseño, la práctica, la implementación, el proceso, el refinamiento, el código.

Desde los inicios de la arquitectura de software, se observó que en la práctica del diseño y la implementación ciertas regularidades de configuración aparecían una y otra vez como respuesta a similares demandas. El número de esas formas no parecía ser muy grande. Muy pronto se las llamó estilos, por analogía con el uso del término en arquitectura de edificios. Un estilo describe entonces una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas. Esas piezas se encuentran repetidamente en la práctica. Una vez que se han identificado los estilos, es lógico y natural pensar en re-utilizarlos en situaciones semejantes que se presenten en el futuro.

Todos los estilos tienen un nombre: cliente-servidor, modelo-vista-controlador, tubería-filtros, arquitectura en capa. Desde que surgieran tanto la disciplina como los estilos no sólo se han generalizado arquitecturas de cliente-servidor, sino que experimentaron su auge primero las configuraciones en varias capas y luego las basadas en componentes, en recursos (la Web) y en servicios (los Web services).

1.6.1 Estilos

1.6.1.1 Puntos a analizar:

1. En arquitectura de software, los estilos surgen de la experiencia que el arquitecto posee; de ningún modo vienen impuestos de manera explícita en lo que elcliente le pide.

2. Los paradigmas como la arquitectura orientada a objetos (p.ej. CORBA), a componentes (COM, JavaBeans, EJB, CORBA Component Model) o a servicios, se relacionan con tecnologías particulares de implementación, un elemento de juicio que en el campo de los estilos se trata a un nivel más genérico y distante, estos paradigmas tipifican más bien como sub-estilos de estilos más englobantes (peer-to-peer, distribuidos, etc)

3. Los patrones arquitectónicos, por su parte, se han materializado con referencia a lenguajes y paradigmas también específicos de desarrollo, mientras que ningún estilo presupone o establece preceptivas al respecto. Si hay algún código en las inmediaciones de un estilo, será código del lenguaje de descripción arquitectónica o del lenguaje de modelado; de ninguna manera será código de lenguaje de programación. Lo mismo en cuanto a las representaciones visuales: los estilos se describen mediante simples cajas y líneas, mientras que los patrones suelen representarse en UML [Lar03].

1.6.1.2 Catálogos de estilos

❖ ¿Cuántos y cuáles son los estilos?

A través del desarrollo de la Arquitectura de Software se han realizado muchas diferenciaciones de tipos de estilos, vamos a examinar algunas:

En un estudio comparativo de los estilos, Mary Shaw [Shaw94] considera los siguientes:

1. Arquitecturas orientadas a objeto
2. Arquitecturas basadas en estados
3. Arquitecturas de flujo de datos: Arquitecturas de control de realimentación.
4. Arquitecturas de tiempo real
5. Modelo de diseño de descomposición funcional
6. Modelo de diseño orientado por eventos
7. Modelo de diseño de control de procesos

8. Modelo de diseño de tabla de decisión

9. Modelo de diseño de estructura de datos

El mismo año, Mary Shaw, junto con David Garlan [GS94], propone una taxonomía diferente, en la que se entremezclan lo que antes llamaba “arquitecturas” con los “modelos de diseño”:

1. Tubería-filtros

2. Organización de abstracción de datos y orientación a objetos

3. Invocación implícita, basada en eventos

4. Sistemas en capas

5. Repositorios

6. Intérpretes orientados por tablas

7. Procesos distribuidos, ya sea en función de la topología (anillo, estrella, etc) o de los protocolos entre procesos (p. ej. algoritmo de pulsación o heartbeat). Una forma particular de proceso distribuido es, por ejemplo, la arquitectura cliente-servidor.

8. Organizaciones programa principal / subrutina.

9. Arquitecturas de software específicas de dominio

10. Sistemas de transición de estado

11. Sistemas de procesos de control

12. Estilos heterogéneos

Otras clasificaciones de estilos son las siguientes, las cuales pertenecen al grupo de Buschmann, quien llama a los estilos “patrones arquitectónicos”:

1. Del fango a la estructura

- Capas

- Tubería-filtros

- Pizarra

2. Sistemas distribuidos

- Broker (p. ej. CORBA, DCOM, la World Wide Web)

3. Sistemas interactivos

- Model-View-Controller

- Presentation-Abstraction-Control

4. Sistemas adaptables

- Reflection (metanivel que hace al software consciente de sí mismo)

- Microkernel (núcleo de funcionalidad mínima)

En Software Architecture in Practice, un texto fundamental de Bass, Clements y Kazman [BCK98] se proporciona una sistematización de clases de estilo en cinco grupos:

1. Flujo de datos (movimiento de datos, sin control del receptor de lo que viene

"corriente arriba")

- Proceso secuencial por lotes

- Red de flujo de datos

- Tubería-filtros

2. Llamado y retorno (estilo dominado por orden de computación, usualmente con un solo thread de control)

- Programa principal / Subrutinas

- Tipos de dato abstracto

- Objetos

- Cliente-servidor basado en llamadas

- Sistemas en capas

3. Componentes independientes (dominado por patrones de comunicación entre procesos independientes, casi siempre concurrentes)

- Sistemas orientados por eventos

- Procesos de comunicación

4. Centrados en datos (dominado por un almacenamiento central complejo, manipulado por computaciones independientes)

- Repositorio

- Pizarra

5. Máquina virtual (caracterizado por la traducción de una instrucción en alguna

otra)

- Intérprete

Considerando solamente los estilos que contemplan alguna forma de distribución o topología de red, Roy Fielding [Fie00] establece la siguiente taxonomía:

1. Estilos de flujo de datos

1.1. Tubería-filtros

1.2. Tubería-filtros uniforme

2. Estilos de replicación

2.1. Repositorio replicado

2.2. Cache

3. Estilos jerárquicos

3.1. Cliente-servidor

3.2. Sistemas en capas y Cliente-servidor en capas

3.3. Cliente-Servidor sin estado

3.4. Cliente-servidor con cache en cliente

3.5. Cliente-servidor con cache en cliente y servidor sin estado

3.6. Sesión remota

3.7. Acceso a datos remoto

4. Estilos de código móvil

4.1. Máquina virtual

4.2. Evaluación remota

4.3. Código a demanda

4.4. Código a demanda en capas

4.5. Agente móvil

5. Estilos peer-to-peer

5.1. Integración basada en eventos

5.2. C2

5.3. Objetos distribuidos

5.4. Objetos distribuidos brokered

6. Transferencia de estado representacional (REST).

En un documento anónimo compilado por Richard Upchurch, del Departmento de Ciencias de la Computación y Información de la Universidad de Massachusetts en Dartmouth se proporciona una “lista de posibles estilos arquitectónicos”, incluyendo:

1. Programa principal y subrutinas
2. Estilos jerárquicos
3. Orientado a objetos (cliente-servidor)
4. Procesos de comunicación
5. Tubería y filtros
6. Intérpretes
7. Sistemas de bases de datos transaccionales
8. Sistemas de pizarra
9. Software bus
10. Sistemas expertos
11. Paso de mensajes
12. Amo-esclavo
13. Asincrónico / sincrónico paralelo
14. Sistemas de tiempo real
15. Arquitecturas específicas de dominio
16. Sistemas en red, distribuidos
17. Arquitecturas heterogéneas

En 2001, David Garlan [Gar01], uno de los fundadores del campo, proporciona una lista más articulada:

1. Flujo de datos

- Secuencial en lotes
- Red de flujo de datos (tuberías y filtros)
- Bucle de control cerrado

2. Llamada y Retorno

- Programa principal / subrutinas

- Ocultamiento de información (ADT, objeto, cliente/servidor elemental)

3. Procesos interactivos

- Procesos comunicantes

- Sistemas de eventos (invocación implícita, eventos puros)

4. Repositorio Orientado a Datos

- Bases de datos transaccionales (cliente/servidor genuino)

- Pizarra

- Compilador moderno

5. Datos Compartidos

- Documentos compuestos

- Hipertexto

- Fortran COMMON

- Procesos LW

6. Jerárquicos

- En capas (intérpretes)

También señala los inconvenientes de la vida real que complican el terreno de las taxonomías: los estilos casi siempre se usan combinados; cada capa o componente puede ser internamente de un estilo diferente al de la totalidad; muchos estilos se encuentran ligados a dominios específicos, o a líneas de producto particulares.

En los últimos cuatro o cinco años se han realizado esfuerzos para definir los estilos de una manera más formal, empleando ADLs como Aesop o Wright, o notaciones en lenguajes de especificación como Z o lenguajes con semántica formal como CHAM.

1.6.1.3 Descripción de los estilos.

A continuación se describen los estilos más representativos y vigentes. De más está decir que, como se ha visto, la agrupación de estilos y sub-estilos es susceptible de realizarse de múltiples formas, conforme a los criterios que se apliquen en la constitución de los ejemplares.

1. Estilos de Flujo de Datos

Esta familia de estilos enfatiza la reutilización y la modificabilidad. Es apropiada para sistemas que implementan transformaciones de datos en pasos sucesivos. Ejemplares

de la misma serían las arquitecturas de tubería-filtros y las de proceso secuencial en lote.

Tubería y filtros

El sistema tubería-filtros se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada. Los datos entran al sistema y fluyen a través de los componentes. En el estilo secuencial por lotes (batch sequential) los componentes son programas independientes; el supuesto es que cada paso se ejecuta hasta completarse antes que se inicie el paso siguiente.

La aplicación típica del estilo es un procesamiento clásico de datos: el cliente hace un requerimiento; el requerimiento se valida; un Web Service toma el objeto de la base de datos; se lo convierte a HTML; se efectúa la representación en pantalla. Otro ejemplo son los workflows.

La documentación establece, que el estilo se debe usar cuando:

- Se puede especificar la secuencia de un número conocido de pasos.
- No se requiere esperar la respuesta asincrónica de cada paso.
- Se busca que todos los componentes situados corriente abajo que sean capaces de inspeccionar y actuar sobre los datos que vienen de corriente arriba (pero no viceversa).

Igualmente, se reconocen como ventajas del estilo tubería-filtros:

- Es simple de entender e implementar. Es posible implementar procesos complejos con editores gráficos de líneas de tuberías o con comandos de línea.
- Fuerza un procesamiento secuencial.
- Es fácil de envolver (wrap) en una transacción atómica.
- Los filtros se pueden empaquetar, y hacer paralelos o distribuidos.

Algunas desventajas son:

- El patrón puede resultar demasiado simplista, especialmente para orquestación de servicios que podrían ramificar la ejecución de la lógica de negocios de formas complicadas.
- No maneja con demasiada eficiencia construcciones condicionales, bucles y otras lógicas de control de flujo. Agregar un paso suplementario afecta la performance de cada ejecución de la tubería.

- Una desventaja adicional referida en la literatura sobre estilos concierne a que eventualmente pueden llegar a requerirse buffers de tamaño indefinido, por ejemplo en las tuberías de clasificación de datos.
- El estilo no es apto para manejar situaciones interactivas, sobre todo cuando se requieren actualizaciones incrementales de la representación en pantalla.
- La independencia de los filtros implica que es muy posible la duplicación de funciones de preparación que son efectuadas por otros filtros (por ejemplo, el control de corrección de un objeto de fecha).

2. Estilos Centrados en Datos

Esta familia de estilos enfatiza la integrabilidad de los datos. Se estima apropiada para sistemas que se fundan en acceso y actualización de datos en estructuras de almacenamiento. Sub-estilos característicos de la familia serían los repositorios, las bases de datos, las arquitecturas basadas en hipertextos y las arquitecturas de pizarra.

- Arquitecturas de Pizarra o Repositorio

En esta arquitectura hay dos componentes principales: una estructura de datos que representa el estado actual y una colección de componentes independientes que operan sobre él [SG96]. En base a esta distinción se han definidos dos subcategorías principales del estilo:

- ✓ Si los tipos de transacciones en el flujo de entrada definen los procesos a ejecutar, el repositorio puede ser una base de datos tradicional (implícitamente no cliente-servidor).
- ✓ Si el estado actual de la estructura de datos dispara los procesos a ejecutar, el repositorio es lo que se llama una pizarra pura o un tablero de control.

Estos sistemas se han usado en aplicaciones que requieren complejas interpretaciones de proceso de señales (reconocimiento de patrones, reconocimiento de habla, etc), o en sistemas que involucran acceso compartido a datos con agentes débilmente acoplados.

También se han implementado estilos de este tipo en procesos en lotes de base de datos y ambientes de programación organizados como colecciones de herramientas en torno a un repositorio común. Muchos más sistemas de los que se cree están organizados como repositorios: bibliotecas de componentes reutilizables, grandes bases de datos y motores de búsqueda. Algunas arquitecturas de compiladores que

suelen presentarse como representativas del estilo tubería-filtros, se podrían representar mejor como propias del estilo de pizarra, dado que muchos compiladores contemporáneos operan en base a información compartida tal como tablas de símbolos, árboles sintácticos abstractos (AST), etcétera. El documento clásico que describe el estilo, Blackboard Systems, de H. Penny Nii [Nii86], bien conocido en Inteligencia Artificial, es en rigor anterior en seis años al surgimiento de la idea de estilos en arquitectura de software. Los estilos de pizarra no son sólo una curiosidad histórica; por el contrario, se los utiliza en exploraciones recientes de inteligencia artificial distribuida o cooperativa, en robótica, en modelos multi-agentes, en programación evolutiva, en gramáticas complejas, en modelos de crecimiento afines a los L-Systems de Lindenmayer, etc.

Un sistema de pizarra se implementa para resolver problemas en los cuales las entidades individuales se manifiestan incapaces de aproximarse a una solución, o para los que no existe una solución analítica, o para los que sí existe pero es inviable por la dimensión del espacio de búsqueda. Todo modelo de este tipo consiste en las siguientes tres partes:

- Fuentes de conocimiento, necesarias para resolver el problema.

- Una pizarra que representa el estado actual de la resolución del problema.

- Una estrategia, que regula el orden en que operan las fuentes.

Las fuentes tratan de resolver el problema cambiando el estado. La única forma en que se comunican entre sí es a través de la pizarra. Finalmente, si de la cooperación resulta una solución adecuada, ésta aparece en la pizarra como paso final.

En un desarrollo colateral, la relación entre este modelo de resolución de problemas y los lenguajes formales fue establecida hacia 1989 por los húngaros

E. Csuhaj-Varjú y J. Kelemen. En su esquema, las fuentes de conocimiento corresponden a gramáticas, el cambio del estado de la pizarra a la re-escritura de formas secuenciales, y la estrategia es representada por modelos de derivación de modo que la solución corresponda a una frase terminal. Estas correspondencias han sido estudiadas primariamente en modelos de programación evolutiva, modelos ecológicos, ecogramáticas, sistemas emergentes, caos determinista, algoritmos genéticos y vida artificial, y en el desarrollo de meta-heurísticas del tipo de la simulación de templado o la búsqueda tabú, temas todos que son demasiado especializados para tratar detalladamente en este contexto [Rey04a]. Últimamente se está hablando mucho de agentes distribuidos, pero más en el sentido de entidades

distribuidas en los cánones de la programación orientada a objetos, que en el de los sistemas complejos y emergentes.

3. Estilos de Llamada y Retorno

Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala. Miembros de la familia son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas.

Model-View-Controller (MVC)

El patrón conocido como Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes [Bur92]:

- Modelo. El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

- Vista. Maneja la visualización de la información.

- Controlador. Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual. La separación entre vista y controlador puede ser secundaria en aplicaciones y, de hecho, muchos frameworks de interfaz implementan ambos roles en un solo objeto. En aplicaciones de Web, por otra parte, la separación entre la vista (el browser) y el controlador (los componentes del lado del servidor que manejan los requerimientos de HTTP) está mucho más taxativamente definida.

Entre las ventajas del estilo señaladas en la documentación de Patterns & Practices de Microsoft están las siguientes:

- ✓ Soporte de vistas múltiples. Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación de Web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes.

- ✓ Adaptación al cambio. Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.
- ✓ Este patrón sentó las bases para especializaciones ulteriores, tales como Page Controller y Front Controller.

Entre las desventajas, se han señalado:

- ✓ Complejidad. El patrón introduce nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad de la solución. También se profundiza la orientación a eventos del código de la interfaz de usuario, que puede llegar a ser difícil de depurar. En rigor, la configuración basada en eventos de dicha interfaz corresponde a un estilo particular (arquitectura basada en eventos) que aquí se examina por separado.
- ✓ Costo de actualizaciones frecuentes. Desacoplar el modelo de la vista no significa que los desarrolladores del modelo puedan ignorar la naturaleza de las vistas.

Arquitecturas en Capas

En [GS94] Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. El número mínimo de capas es obviamente dos, y en ese umbral la literatura arquitectónica sitúa a veces al sub-estilo cliente-servidor como el modelo arquetípico del estilo de capas y el que se encuentra con mayor frecuencia en las aplicaciones en red: Un componente servidor, que ofrece ciertos servicios, escucha que algún otro componente requiera uno; un componente cliente solicita ese servicio al servidor a través de un conector. El servidor ejecuta el requerimiento (o lo rechaza) y devuelve una respuesta.

Las ventajas del estilo en capas son obvias:

- ✓ El estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- ✓ El estilo admite muy naturalmente optimizaciones y refinamientos.

- ✓ Proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.

También se han señalado algunas desventajas de este estilo.

Muchos problemas no admiten un buen mapeo en una estructura jerárquica. Incluso cuando un sistema se puede establecer lógicamente en capas, consideraciones de performance pueden requerir acoplamientos específicos entre capas de alto y bajo nivel. A veces es también extremadamente difícil encontrar el nivel de abstracción correcto. Además, los cambios en las capas de bajo nivel tienden a filtrarse hacia las de alto nivel, en especial si se utiliza una modalidad relajada; también se admite que la arquitectura en capas ayuda a controlar y encapsular aplicaciones complejas, pero complica no siempre razonablemente las aplicaciones simples.

Arquitecturas Basadas en Componentes

Hay un buen número de definiciones de componentes, pero Clemens Alden Szyperski proporciona una que es bastante operativa: un componente de software, dice, es una unidad de composición con interfaces especificadas contractualmente y dependencias del contexto explícitas [Szy02]. Que sea una unidad de composicióny no de construcción quiere decir que no es preciso confeccionarla: se puede comprar hecha, o se puede producir en casa para que otras aplicaciones de la empresa la utilicen en sus propias composiciones. Pragmáticamente se puede también definir un componente como un artefacto diseñado y desarrollado de acuerdo ya sea con CORBA Component Model (CCM), JavaBeans y Enterprise JavaBeans en J2EE y lo que alternativamente se llamó OLE, COM, ActiveX y COM+, y luego .NET.
En un estilo de este tipo:

- ✓ Los componentes en el sentido estilístico son las unidades de modelado, diseño e implementación.
- ✓ Las interfaces están separadas de las implementaciones, y las interfaces y sus interacciones son el centro de incumbencias en el diseño arquitectónico.
- ✓ Los componentes soportan algún régimen de introspección, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución.

- ✓ En cuanto a las restricciones, puede admitirse que una interfaz sea implementada por múltiples componentes. Usualmente, los estados de un componente no son accesibles desde el exterior [Szy02]. Que los componentes sean locales o distribuidos es transparente en la tecnología actual.

El marco arquitectónico estándar para la tecnología de componentes está constituido por los cinco puntos de vista de RM-ODP (Empresa, Información, Computación, Ingeniería y Tecnología). La evaluación dominante del estilo de componentes subraya su mayor versatilidad respecto del modelo de objetos, pero también su menor adaptabilidad comparado con el estilo orientado a servicios.

Las tecnologías de componentes del período de inmadurez, asimismo, se consideraban afectadas por problemas de incompatibilidad de versiones e inestabilidad que ya han sido largamente superados en toda la industria.

Ventajas de utilizar los Estilos

Lo más importante de los estilos es que ofrecen evidencia basada en la experiencia sobre cómo se ha utilizado cada clase históricamente, junto con razonamiento cualitativo para explicar por qué cada clase posee propiedades específicas. Los estilos son entonces una poderosa herramienta para el reutilizador porque proporcionan una sabiduría decantada por muchos diseñadores precedentes que afrontaron problemas similares.

Aquí se enumeran algunas ventajas de utilizar estilos:

1. Los estilos promueven reutilización de diseño. Las soluciones de rutina con propiedades bien entendidas se pueden aplicar otra vez a nuevos problemas con alguna confianza.

2. El uso de estilos puede conducir a una significativa reutilización de código. Los aspectos invariantes de un estilo conducen a replicar implementaciones.

3. Es más fácil para otros entender la organización de un sistema si se utilizan estructuras convencionales. Por ejemplo, la tipificación de un sistema como “cliente-servidor” de inmediato evoca una fuerte imagen respecto a cuales son sus piezas y cómo se vinculan recíprocamente.

4. Al acotar el espacio de diseño, un estilo arquitectónico permite de inmediato análisis especializados específicos del estilo.

5. Es usualmente posible, e incluso deseable, proporcionar visualizaciones específicas de un estilo. Esto proporciona representaciones gráficas y textuales que coinciden con

intuiciones específicas de dominio sobre cómo deben representarse los diseños en él [GKM+96].

6. El uso de estilos se orienta a reducir costos de desarrollo de aplicaciones y aumentar el potencial para la comunalidad entre diferentes miembros de una familia estrechamente relacionada de productos.

Patrones

Existen diversas clases de patrones: de análisis, de arquitectura, de diseño, de organización o proceso y los llamados idiomas.

La principal diferencia entre estilos y patrones es que un estilo representa específicamente una familia arquitectónica, construida a partir de bloques de construcción arquitectónicos, tales como los componentes y los conectores. Los patrones, en cambio, atraviesan diferentes niveles de abstracción y etapas del ciclo de vida partiendo del análisis del dominio, pasando por la arquitectura de software y yendo hacia abajo hasta el nivel de los lenguajes de programación [10].

Por lo antes descrito, más la experiencia que existe en los proyectos productivos de nuestra Universidad y además la facilidad de integración y optimización de tiempo utilizaremos el estilo n-capas específicamente el patrón de 3 capas.

Conclusiones

En este capítulo se hizo un análisis de diferentes sistemas existentes tanto en el mundo, en nuestro país, como en la Universidad de las Ciencias Informáticas (UCI), lo que demostró la conveniencia de asumir el desarrollo de una aplicación, a partir de la insuficiencia de las existentes para dar respuesta a los requerimientos que se plantean. Luego de hacer una investigación a profundidad se mostró el por qué se determinaron las diferentes herramientas a utilizar en el desarrollo del proyecto, además de la metodología a utilizar, así como los lenguajes que se utilizarán tanto para modelar la aplicación como para su implementación. Tomando en cuenta estas precisiones pasaremos a mostrar los elementos del Capítulo 2. En el mismo se mostrarán: Elementos fundamentales del Negocio así como del Sistema, una descripción más detallada de los Casos de Uso tanto del Negocio como del Sistema, que en este caso son los mismos. Se describirán además todos los Casos de Uso siguiendo los diagramas de Actividades o viceversa, lo que nos dará una Idea más clara de la solución del problema a resolver. Conoceremos además los requerimientos tanto funcionales como no funcionales, los que nos darán una guía de todo lo que debemos hacer para satisfacer las necesidades planteadas [11].

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

2. Introducción

El presente capítulo hace un análisis del negocio en cuanto a su funcionamiento, documentos y sistemas necesarios para llevarlo a cabo. En el mismo se identifican las necesidades del cliente y se hace un estudio del entorno de trabajo. Además se hace una propuesta de sistema a partir del resultado del análisis de los requerimientos tanto funcionales como no funcionales.

2.1 Procesos de Negocio.

El proceso de obtención de la variación del rating Elo es muy importante para el desarrollo del Ajedrez de alto rendimiento en la UCI, lo que constituye una función importante de la Cátedra de Ajedrez, la que actualmente no cuenta con una herramienta que de forma eficaz, rápida e integrada, haga los cálculos que se requiere y brinde las informaciones necesarias con un mínimo de trabajo. Se hace notar que en los inicios del trabajo de la Cátedra, se comenzó a llevar un rating Elo local, de forma manual, y fue necesario desistir de esta prestación, debido a la carga de trabajo que representaba para el trabajo de la Cátedra, y el hecho de que en la práctica, se acumuló un atraso en la actualización, que se hizo insalvable. Desde entonces quedó claro, que sólo podría considerarse el establecimiento de un rating Elo en la UCI, cuando pudiera disponerse de una Aplicación Web que se ocupara de llevarlo de forma automática.

2.2 Objeto de Automatización

Serán objeto de automatización las funciones relacionadas con el proceso de Cálculo de Variación de puntos Elo, obteniendo información importante sobre los ajedrecistas de la Universidad y todo aquel que esté vinculado a torneos realizados en la misma. Además, será objeto de automatización el trabajo de actualización de los estados del Elo para cada usuario.

En estos momentos no se cuenta con una aplicación ni de escritorio ni Web que se encargue del trabajo relativo al Cálculo del Elo el cual es la fuente de información para los reportes que se obtendrán a partir de un futuro y actualizado escalafón de jugadores.

En la UCI en estos momentos los datos de estos ajedrecistas no se encuentran en una base de datos específica, por lo que nos corresponde construir la misma. Esto

facilitaría el proceso de obtener los datos que el sistema propuesto necesita para dar respuesta a los requerimientos que el cliente planteó. Podríamos utilizar en un futuro una Base de Datos común, que estaría relacionada con una o varias de las ya existentes que a continuación se enuncian:

Akademos: Para la obtención referente a los estudiantes de la Universidad.
Trabajadores: Para la información referente tanto a los profesores como a los trabajadores con que cuenta la Universidad vinculados al ajedrez.
Directorio: En esta base de datos se encuentra la información común para cada persona de la UCI.
Identificación: Esta base de datos contiene las fotos y el número de credencial de todas las personas de la UCI.

2.3 Propuesta del Negocio.

El módulo del Cálculo de la Variación del Rating Elo permitirá obtener información confiable sobre determinados aspectos requeridos. Lo anteriormente expresado está basado en el hecho de que el sistema será desarrollado como una aplicación Web, lo cual permite que pueda ser utilizado en cualquier momento desde cualquier lugar, dentro de la UCI, siempre que se tenga permiso de acceso a las funcionalidades que el sistema brinda. Con la implementación y puesta en funcionamiento del sistema se podrá hacer un mejor manejo de la información que se genera; pero el primer paso para lograr una aplicación que responda a los requerimientos del cliente es comprender el negocio actual dentro de la organización. Con este objetivo se lleva a cabo el Modelo de Negocio, el cual hace un levantamiento de cómo funcionan los procesos hasta el momento. A continuación se mostrarán descripciones textuales y diagramas que permitirán un mejor entendimiento del Negocio.

2.3.1 Determinación y Justificación de los Actores y Trabajadores

Tabla 1. Definición de actores del negocio.

Actores del Negocio	Justificación
Módulo Arbitraje	Entrega los resultados del Torneo.
Reloj	Finalizado el Torneo se muestran los reportes.

Tabla 2.Definicion de Trabajadores del Negocio.

Trabajadores del negocio	Justificación
Arbitro	Encargado de Calcular la Variación de Rating Elo y Sistema de desempates además de mostrar los reportes en cada torneo.

2.3.2 Diagrama de Casos de Uso del Negocio

El diagrama de casos de uso del negocio tiene una gran importancia porque representa cómo interactúan los actores y casos de uso del negocio, ayudando al equipo de desarrollo a comprender cómo funciona el proceso de negocio con la consiguiente ayuda a la hora de obtener los casos de uso del sistema.

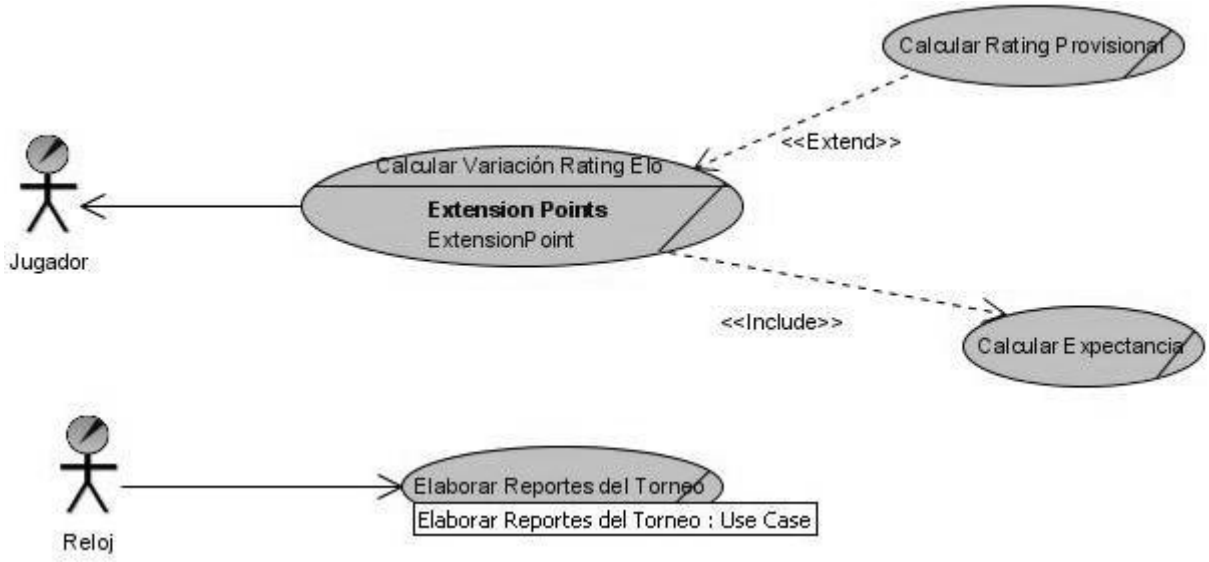


Figura 2.1 Diagrama de Casos de Uso del Negocio.

2.3.3 Descripción de Casos de Uso del Negocio

Tabla 3. Descripción del caso de uso del negocio: Variación Rating Elo.

CU-1	Variación Rating Elo
Actor	Jugador

Trabajador	Arbitro
Descripción	Se le calcula la variación del Rating a cada jugador del torneo luego de terminado el torneo según la fórmula obteniendo como datos esenciales el rating elo inicial, la constante k, los puntos alcanzados al culminar el torneo y la expectancia.

Tabla 4. Descripción del caso de uso del negocio: Enviar Reportes.

CU-2	Enviar Reportes
Trabajador	Arbitro
Descripción	Se muestran los resultados del torneo, es decir los lugares en los que quedó cada jugador y la variación del rating elo, siempre al finalizar el torneo.

Nota: Esta es una breve descripción de los casos de uso fundamentales de este negocio, los cuales incluyen otros casos de uso, los que serán descritos detalladamente a continuación, para una mejor comprensión del sistema a realizar, se describirán paso por paso según lo que debe hacerse en cada caso, será una especificación de los Diagramas de actividades de manera que tengamos un primer acercamiento a las funciones que debe cumplir nuestra Aplicación.

2.3.3.1 Descripción Detallada del Caso de Uso Variación Rating Elo.

Tabla 5. Descripción detallada del caso de Uso Variación Rating Elo

Caso de Uso :	Calcular la Variación Rating Elo
Actores :	Módulo Arbitraje
Trabajadores :	Arbitro
Propósito :	Calcular el Rating Elo de todos los Jugadores del Torneo.

Resumen :	El caso de Uso se inicia al terminar un torneo. El árbitro tiene el registro de los resultados que se necesitan para iniciar el proceso. Luego verifica si hay jugadores UR, en caso de haber le calcula el Rating Elo Provisional a cada uno de los Jugadores U/R, y una vez calculado éste, pasa al proceso común, es decir calcula expectancia, toma los resultados de cada jugador, su Rating inicial, la constante K, y calcula el nuevo Rating_Elo de todos los jugadores participantes en el torneo y muestra los resultados.
Casos de Uso Asociados :	**Cálculo Expectancia, Cálculo Rating Provisional.**
Flujos de Trabajo :	
Acción del Actor	**Respuesta del Negocio**
1. Solicita los Resultados de cada Jugador.	2. El árbitro verifica si hay Jugadores UR. 3. El árbitro calcula el rating Elo inicial provisional de todos los jugadores U/R. 4. El árbitro calcula la Expectancia de todos los jugadores. 5. El árbitro toma los puntos válidos para la Variación de Rating de todos los jugadores. 6. El árbitro obtiene la constante K de cada Jugador, de la base de datos. 7. El árbitro Calcula la variación del Rating_Elo de todos los jugadores . 8. El árbitro visualiza los resultados.
Prioridad :	Alta

Cursos Alternos	
Acción del Actor	**Respuesta del Negocio**
	2.1) Si el árbitro verifica que hay jugadores UR entonces pasa a Calcular el Rating_Provisional.
	2.2) El árbitro Calcula el Elo Inicial de estos jugadores.
	2.4) Se va a la Acción 3 del Flujo Normal de los Eventos.

2.3.3.2 Descripción Detallada del Caso de Uso Mostrar Reportes

Tabla 6. Descripción detallada del caso de Uso Enviar Reportes

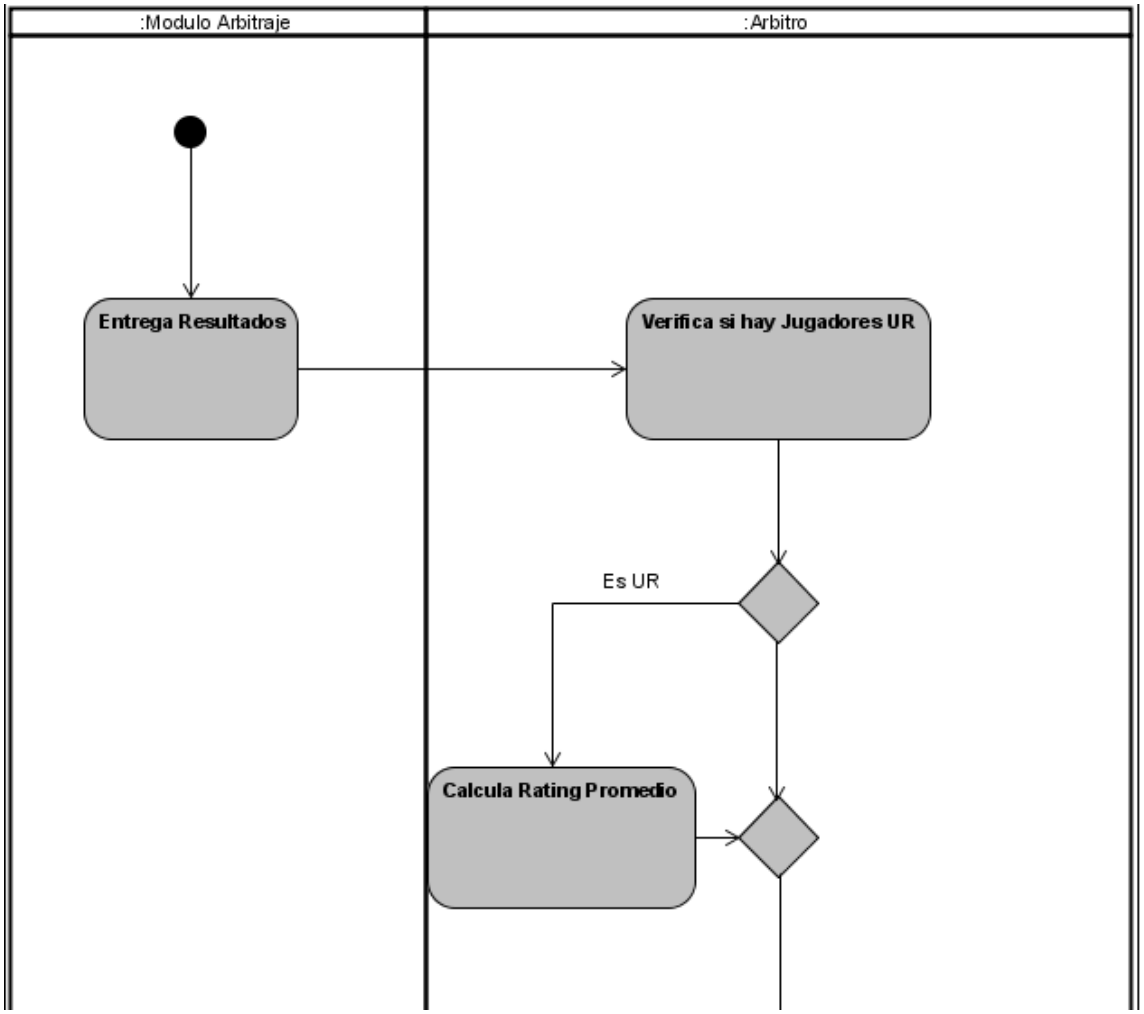
Caso de Uso :	Enviar Reporte
Actores :	Reloj
Trabajadores :	Arbitro
Propósito :	Enviar Reporte
Resumen :	El caso de Uso se inicia al terminar un torneo. El árbitro obtiene los resultados, determina los lugares de cada Jugador aplicando los sistemas de Desempate establecidos, y muestra los resultados.
Casos de Uso Asociados :	**Lugares en el Torneo, Sistema de Desempate.**
Acción del Actor	**Respuesta del Negocio**
1. Terminar el torneo.	2. El árbitro obtiene los resultados (Puntos acumulados por cada Jugador). 3. Verifica que no hay empates en los Lugares a partir de los puntos obtenidos por cada jugador. 4. El árbitro visualiza los Lugares de cada jugador en el torneo.
Prioridad:	Alta

Cursos Alternos	
Acción del Actor	**Respuesta del Negocio**
	3.1) Si el árbitro verifica que hay empate, entonces pasa a aplicar el sistema de desempate (los que definen nuevas puntuaciones destinadas a obtener resultados del torneo con coeficientes individuales diferentes).
	3.2) Se va a la Acción 4 del Flujo Normal de los Eventos.

2.3.4 Diagramas de Actividades por casos de uso

Los diagramas de actividades permiten ver el flujo de actividades que se lleva a cabo; así como la descripción textual de cada caso de uso permitiendo la identificación de las actividades objeto de automatización. En este caso las actividades objeto de automatización están señaladas en el diagrama, mostrando a continuación las descripciones detalladas de cada caso de uso. De aquí una mejor visión del proyecto, describiendo los pasos a seguir para obtener resultados óptimos, reconoce los casos de uso que pasan al sistema siguiendo la estrategia de describirlos a continuación sobre las tablas, muestra las esperas paralelas de acciones que deben esperar por otras para poder continuar con la realización de la solución del problema a resolver.

Los diagramas de actividades por casos de uso prueban la ventaja de descripción clara y detallada que hace que un programador comprenda realmente el negocio, digo programador como cliente y personal que se incentive a investigar sobre cualquier tema a tratar, en este caso específico el del Cálculo de la Variación del Rating Elo en la Universidad de las Ciencias Informáticas con una vista futura o alcance nacional y en algún momento probado, internacional.


:Modulo Arbitraje
:Arbitro
Entrega Resultados
Verifica si hay Jugadores UR
Es UR
Calcula Rating Promedio

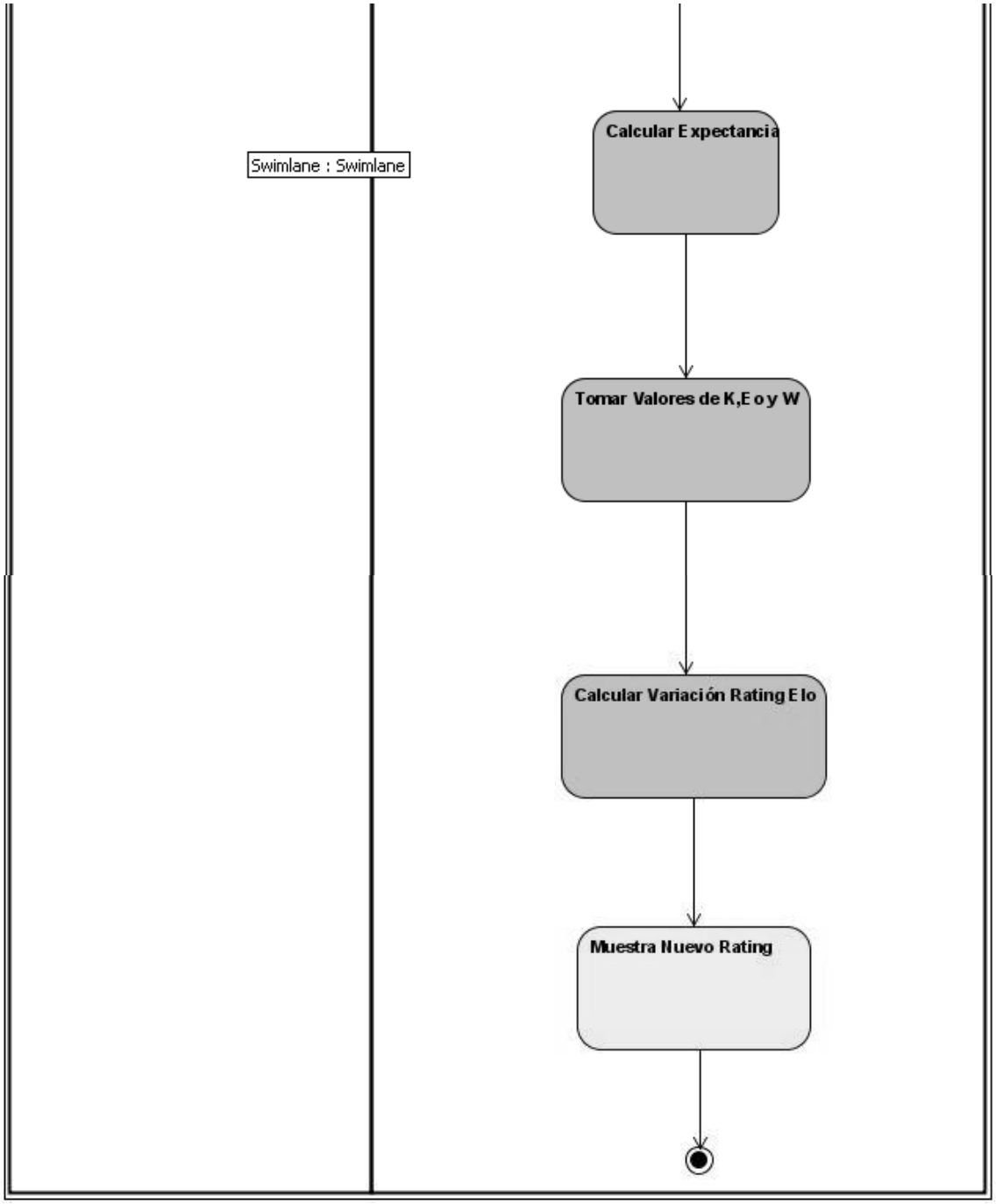


Figura 2.3 Diagrama de Actividades del Caso de Uso Calcular Variación Rating Elo

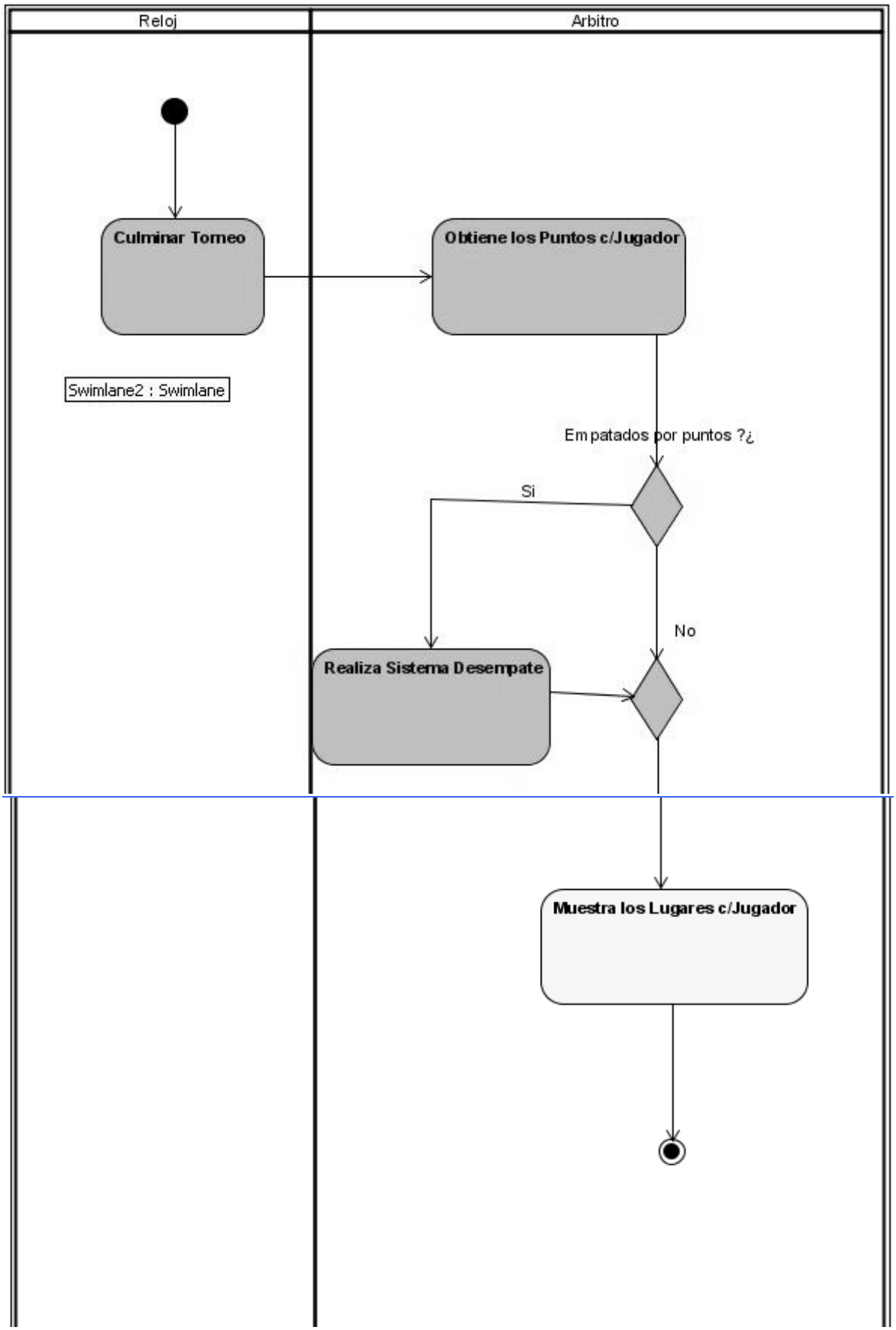


Figura 2.4 Diagrama de Actividades del Caso de Uso Enviar Reportes:

2.4 Especificación de los requisitos de Software

Los requisitos de un software es específicamente lo que se necesita para construirlo, de vital importancia en el desarrollo del mismo ya que pasa a ser la guía de cada unos de los procedimientos a seguir, están los funcionales y los no funcionales.

Funcionales: Lo que no puede dejar de tener el software, lo ideal, lo central, lo específico, lo que realmente necesita el cliente.

No Funcionales: La portabilidad, seguridad, Apariencia o interfaz externa, Rendimiento, Soporte, entre otros, lo adicional del sitio, no menos importante por mejorar estos su calidad.

A continuación mostramos los requisitos tanto funcionales como no funcionales, según quedaron establecidos para nuestra Aplicación.

2.4.1 Requerimientos Funcionales.

Aplicación Web

1. Mostrar resultados de la variación del Cálculo del Rating Elo.
    - 1.1 Obtener Elo Inicial en la Base de Datos para cada Ajedrecista.
    - 1.2 Obtener la Constante K en la Base de Datos para cada Ajedrecista.
    - 1.3 Calcular los Puntos alcanzados por cada Ajedrecista al finalizar el Torneo.
    - 1.4 Calcular Rating Provisional inicial de cada Jugador UR.
    - 1.5 Calcular Expectancia de cada Ajedrecista al iniciarse un torneo.
2. Mostrar Reporte de Lugares en el Torneo.
    - 2.1 Obtener los resultados de los Jugadores en el torneo.
    - 2.2 Mostrar los jugadores en orden por cantidad de puntos obtenidos.
        - 2.2.1 Si hay empate aplicar Sistema de Desempate.

2.4.2 Requerimientos no Funcionales

2.4.2.1 Apariencia o Interfaz Externa

- ✓ La interfaz tendrá un diseño agradable e intuitivo.

2.4.2.2 Usabilidad

- ✓ El sistema en general se desarrolla con el objetivo de facilitar el trabajo realizado hasta el momento y mantener almacenada toda la información.

- ✓ La interfaz mostrará únicamente funcionalidades referentes a las tareas para las que está destinado el sistema.
- ✓ Se garantiza que los usuarios sólo tengan acceso a la información que les corresponda.
- ✓ Facilidad de uso para todo tipo de clientes, incluyendo personas con pocos conocimientos en el uso de las computadoras.

2.4.2.3 Rendimiento

El sistema debe garantizar la mayor eficiencia posible en cuanto al tratamiento de la información, de manera que la velocidad de procesamiento de la misma sea la máxima posible. Se debe garantizar la consistencia y disponibilidad de la información en todo momento, por lo que se requiere además un tiempo de recuperación mínimo.

2.4.2.4 Seguridad

Las páginas del sitio solo estarán accesibles para personas que tengan usuarios autorizados en el servidor. La información que se obtiene no será para conocimiento de cualquier persona (salvo en el caso de información que se considere pública) sino para los interesados por requerimientos de cargo y responsabilidad.

2.4.2.5 Soporte

El sistema debe ser de fácil instalación, adaptable a numerosas plataformas y de fácil mantenimiento.

2.4.2.6 Portabilidad

Facilidad para adaptarlos a diferentes ambientes sin necesidad de usar otros medios que los previstos. Con facilidad de uso en Windows y Linux como plataforma.

2.5 Definición de los Casos de Uso del Sistema

Los casos de uso del sistema serán en definitiva los encargados de guiar todo el proceso de desarrollo del sistema a implementar. Es importante determinar en esta etapa los actores del sistema así como los casos de uso de los cuales se benefician.

2.5.1 Definición de los Actores

Se definen los actores del sistema y se justifica el por qué de la elección.

Tabla 7. Definición de actores del sistema.

Actores	**Justificación**
Jugador	Se le Calcula la Variación del Rating Elo.
Reloj	Marca el tiempo en que se efectuará el reporte.

2.5.2 Listado de Casos de Uso

Se presenta un listado de los casos de uso del sistema con una breve información de quien es el actor que lo inicializa, el nombre del caso de uso y una breve descripción del mismo.

Tabla 8. Definición del CU Calcular_Variación_del_Rating_Elo.

CU-1	Calcular Variación del Rating Elo
Actor	Arbitro
Descripción	El caso de Uso se inicia al terminar un torneo. El módulo de Arbitraje muestra los resultados. Luego el árbitro verifica si hay jugadores UR, en caso de haber le calcula el Rating Elo Provisional, y una vez calculado este pasa al proceso común, es decir calcula expectancia, toma los resultados de cada jugador, su Rating inicial, la constante K, el árbitro Calcula el nuevo Rating_Elo de todos los jugadores participantes en el torneo y muestra los resultados.
Referencia	

Tabla 9. Definición del CU Enviar_Reporte

CU-2	Enviar Reporte
Actor	Reloj

Descripción	El caso de Uso se inicia al terminar un torneo. El árbitro obtiene los resultados, verifica que no haya empate, muestra los resultados.
Referencia	

2.5.3 Diagrama de casos de uso del sistema

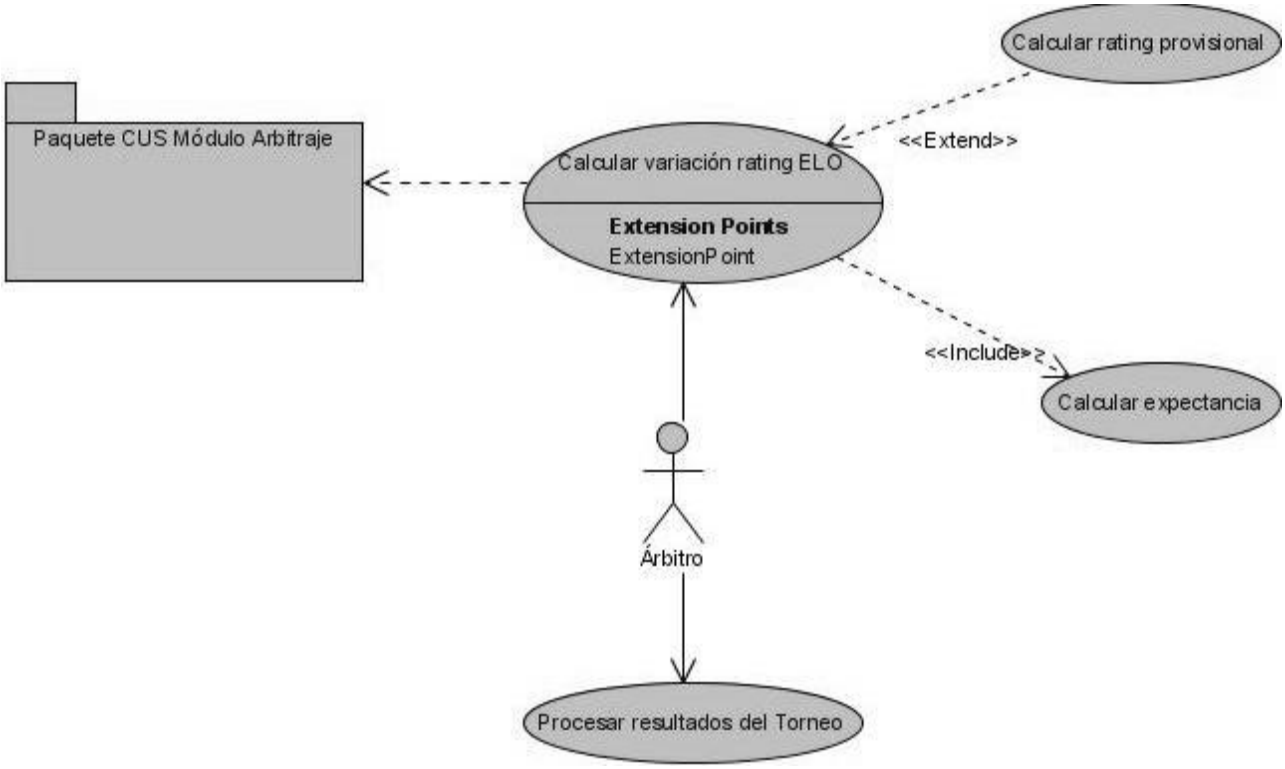


Figura 2.5 Diagrama de Casos de Uso del Sistema

2.5.4 Casos de Uso Expandidos

Tabla 10. Definición del CU Rating_Provisional_inicial

CU-3	Rating Provisional
Actor	
Descripción	El caso de Uso se inicia al verificar que existe un jugador UR. El árbitro calcula el rating promedio de todos los jugadores que se enfrentaron al mismo y calcula la Bonificación o Penalización a través de una Tabla, y la suma que se obtiene pasa a ser su

	Rating Provisional inicial.
Referencia	CU_1 Requisito Funcional 1.4.

La prioridad en este caso de uso es alta debido a que se hace necesario en un torneo donde exista al menos un jugador UR calcularle el Rating Inicial que será el que se tendrá en cuenta a la hora de asignarle su primer valor de Rating Elo.
Este caso de uso expandido no tiene caso de uso asociado debido a que se ejecuta sin tener que acudir a algún otro caso de uso, por su complejidad y lo antes mencionado además del proceso expandido que lleva se ha denominado como caso de uso, quien además se hace transparente en la Implementación de la Aplicación.
Tal como este caso de uso expandido le sigue uno no menos importante necesario para calcular la Variación de Rating Elo.

Tabla 11. Descripción detallada del caso expandido Expectancia

CU-4	Calcular Expectancia
Actor	
Descripción	El caso de Uso se inicia al ya conocer los jugadores que participarán en el torneo, es decir su Rating Elo, se Calcula la diferencia de Elo de un jugador contra su oponente y luego se le Calcula la Expectancia
Referencia	CU_1 Requisito Funcional 1.5.

El caso de uso expandido del sistema Calcular Expectancia es uno de los más importantes entre los casos de uso presentes en el sistema puesto que es una información necesaria para obtener la variación del Rating Elo de cada jugador participante en el torneo. Puede obtenerse al comenzar el torneo o al este ser finalizado, esto se puede hacer en el caso de que el torneo sea Round Robin; quien es el tipo de torneo que estamos implementando, cambiaría en otros tipos de torneo destacando como ejemplo esencial y más complicado el Suizo.

Conclusiones

En este capítulo se hizo un análisis del entorno de trabajo y se identificaron los requerimientos del cliente. Por otra parte se declararon los requerimientos formulados por el cliente así como los no funcionales encontrados tanto por el cliente como por el analista. Como resultado de lo expresado anteriormente se logra una mayor claridad en cómo funciona el negocio y qué se hará para implementar el sistema propuesto.

CAPITULO 3. ANÁLISIS Y DISEÑO DEL SISTEMA

3. Introducción

Este capítulo presenta el análisis y diseño del Subsistema para el Manejo y Reportes del Cálculo del Rating Elo. El mismo mostrará “qué” y “cómo” se implementará dentro de la aplicación. El análisis dará una visión general de qué se debe hacer sin entrar en detalles, mostrando desde tres estereotipos básicos: interfaz, negocio y entidad, la forma aproximada que tendrá el sistema. Mientras que el diseño dará como resultado un “mapa” de la implementación, lo cual será gracias al diagrama de clases del diseño que mostrará las clases necesarias con sus métodos y atributos. Como parte del diseño se mostrarán también los diagramas de secuencia que darán una visión dinámica de las clases colaborando indudablemente a la comprensión de la tarea a desarrollar.

3.1 Diagrama de clases del análisis

El análisis, como se ha expresado previamente, brinda una visión muy general de lo que debe hacer el sistema. El modelo del análisis brinda una vista interna del sistema que está descrita en el lenguaje del desarrollador. A continuación se expone el diagrama de clases del análisis del sistema propuesto.

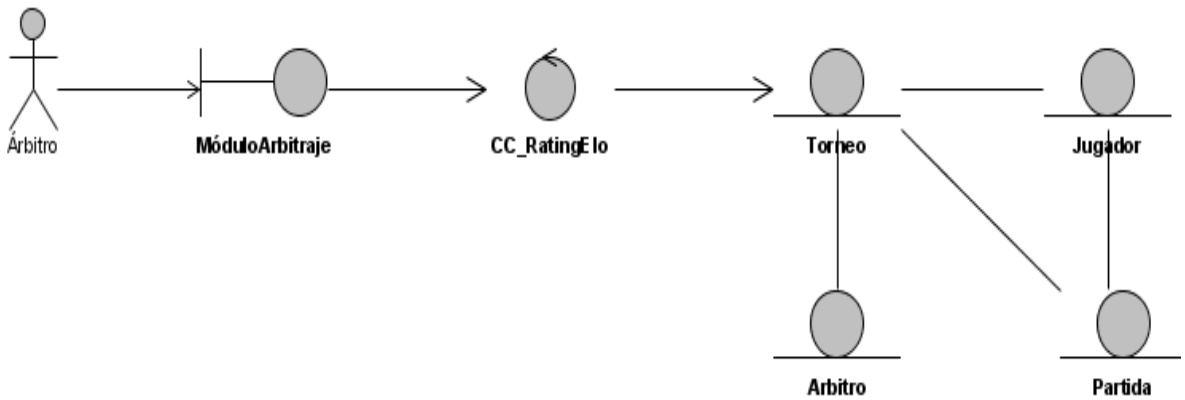


Figura 3.1 Diagrama Clases del Análisis.

3.2 Diseño

Los diagramas de clase del diseño serán representados en esta sección; de los cuales, se mostrarán los diagramas de clases para cada caso de uso. Para una mejor comprensión de las clases representadas en estos diagramas serán descritas de forma textual sus métodos y atributos. Para representar el aspecto dinámico de cada uno de los diagramas de esta sección, serán utilizados los diagramas de secuencia los cuales representan la interacción entre clases. A continuación se representan los diagramas de clases del diseño para cada caso de uso.

3.2.1 Diagramas de Clases del Diseño.

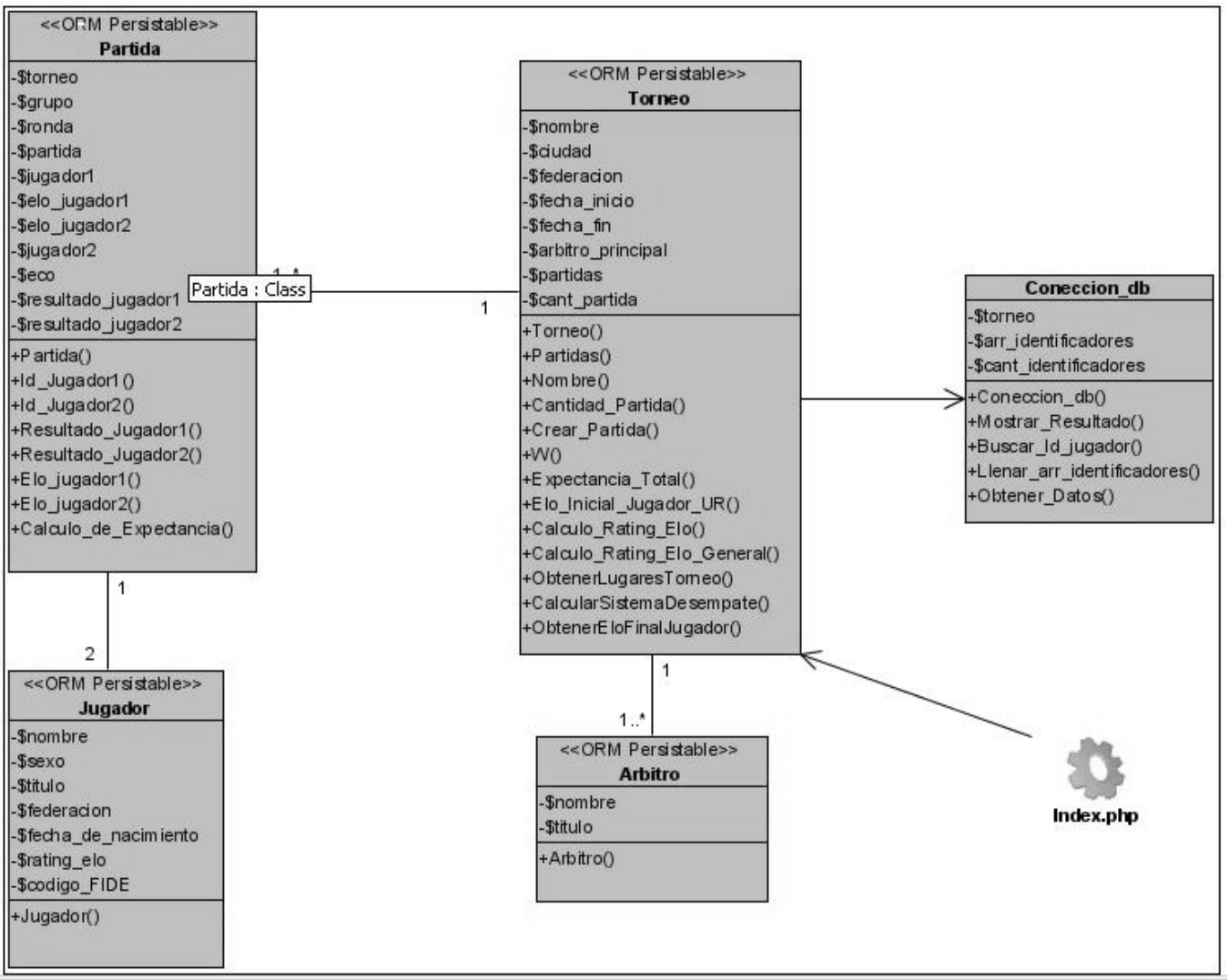


Figura 3.2 Diagramas de Clases del Diseño

3.2.2 Descripción de las clases

A continuación se hace una descripción de las clases del diseño las cuales fueron expuestas anteriormente. Esta sección tiene gran importancia porque describe a un desarrollador que se incorpore nuevo, las clases existentes brindándole información de los atributos y los métodos. Además esclarece a cualquier lector sobre las clases existentes y sus responsabilidades.

Tabla 12. Descripción de la Clase Torneo.

Nombre: Torneo	
Tipo de clase Controladora	
Atributo	**Tipo**
nombre	string
ciudad	string
federacion	string
fecha_inicio	Fecha
fecha_fin	Fecha
arbitro_principal	string
cant_jugadores	int
Para cada responsabilidad:	
Nombre:	**Descripción:**
Torneo	Permite inicializar cada atributo de la clase.
Cant_Partidas	Devuelve la cantidad de partidas que va a tener el torneo.
W	Devuelve los puntos alcanzados por cada jugador al culminar el torneo
Expectancia_Total	Calcula la expectancia de cada jugador en el torneo, es decir la probabilidad que tiene de ganar o perder al comenzar el torneo contra otro jugador.
Elo_Inicial_Jugador_UR	Calcula el promedio de los jugadores oponentes del jugador al que se necesita calcular el Rating Elo, y el resultado se le asigna al mismo.
Calcular_Rating_Elo_General	Calcula el Rating Elo final de todos los jugadores.

Tabla 13. Descripción de la Clase Partida

Nombre: Partida	
Tipo de clase: Modelo	
Atributo	**Tipo**
torneo	string
grupo	string
ronda	int
partida	int
Jugador1	string
Jugador2	string
Resultadojugador1	float
Resultadojugador2	float
Para cada responsabilidad:	
Nombre:	**Descripción:**
Partida	Permite inicializar cada atributo de esta clase.
Id_Jugador1	Devuelve el identificador del jugador1.
Id_Jugador2	Devuelve el identificador del jugador2.
Resultado_Jugador1	Retorna el resultado del Jugador 1 en la partida.
Resultado_Jugador2	Retorna el resultado del Jugador 2 en la partida.
Calcular_Expectancia	Permite conocer la expectancia de cada jugador al celebrarse cada partida en cada ronda.

Tabla 14. Descripción de la Clase Jugador

Nombre: Jugador	
Tipo de clase Modelo	
Atributo	**Tipo**
nombre	string
sexo	string
titulo	string
federacion	string
Fecha_nacimiento	Fecha
Rating_Elo	int
código_Fide	string
Para cada responsabilidad:	
Nombre:	**Descripción:**
Jugador	Permite Inicializar cada atributo de la clase.

Tabla 15. Descripción de la clase Arbitro.

Nombre: Arbitro	
Tipo de clase Modelo	
Atributo	**Tipo**
nombre	string
Para cada responsabilidad:	
Nombre:	**Descripción:**
Arbitro	Permite inicializar cada dato del Arbitro

Tabla16. Descripción de la clase Conección_DB

Nombre: Coneccion_DB	
Tipo de clase Modelo	
Atributo	**Tipo**
torneo	string
Arr_ identificadores	arreglo
Cant_identificadores	int
Para cada responsabilidad:	
Nombre:	**Descripción:**
Coneccion_DB	Permite encontrar todos los atributos en la conección de la base de batos.
Mostrar_Resultados	Muestra los resultados de cada jugador del torneo
Mostrar_Id_Jugador	Busca el jugador y muestra sus datos.
Llenar_Arr_Identificadores	Permite Modificar los datos de los jugadores en torneo.
Obtener Datos	Obtiene los datos de los jugadores en torneo.

3.2.3 Diagrama de Clases Persistentes

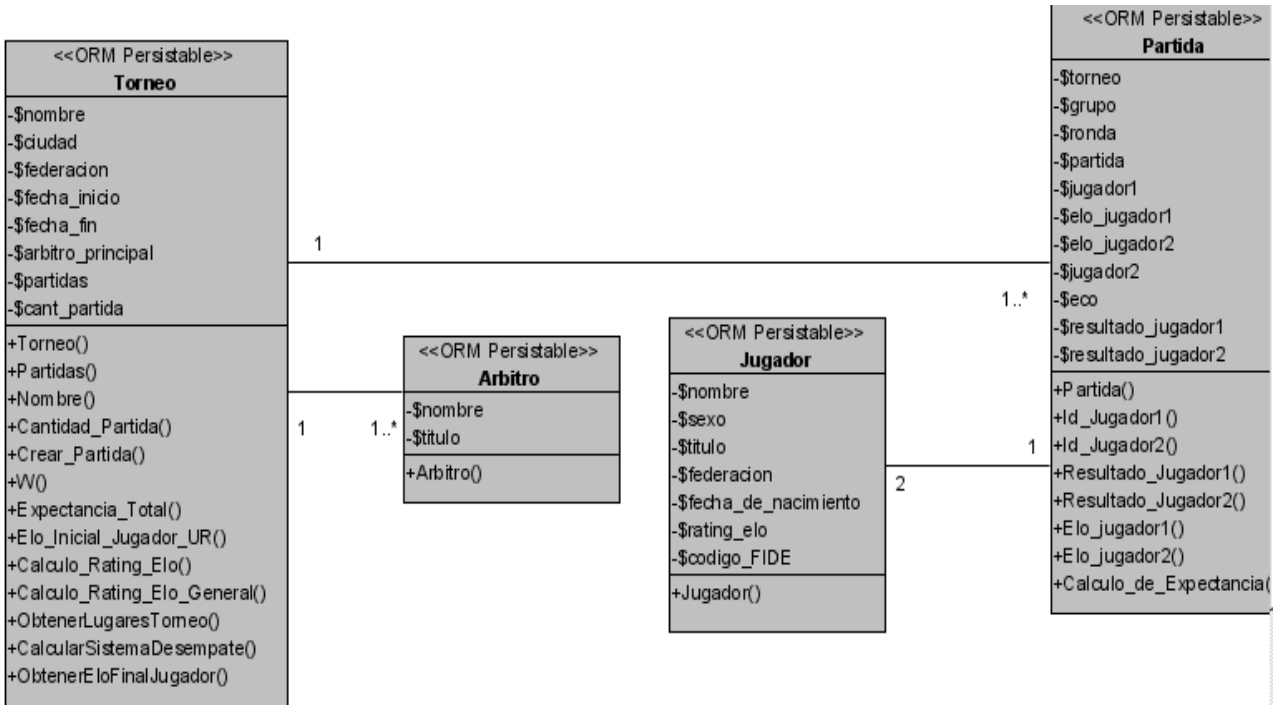


Figura 3.3 Diagrama de Clases Persistentes

3.3.1 Descripción de cada tabla de la Base de Datos.

Tabla 17. Descripción de la Tabla Torneo.

Nombre: torneo		
Descripción: Esta tabla contiene los datos de un torneo.		
Atributo	**Tipo**	**Descripción**
Identificador	int	Llave primaria que contiene el identificador de cada torneo
Nombre	varchar	Contiene el nombre de cada torneo.
Idciudad	int	Llave que contiene el identificador de la ciudad donde ce celebra el torneo.

Idfederación	varchar	Llave que contiene el identificador de la federación a la que pertenece.
FechaInicio	datetime	Este campo contiene la fecha en que comienza el torneo
FechaFin	datetime	Este campo contiene la fecha de expiración del torneo.
Id_arbitro_principal	int	Llave que contiene el identificador del árbitro principal actuante en el torneo.

Tabla 18. Descripción de la Tabla Torneo_Ronda

Nombre: torneo_ronda		
Descripción: Esta tabla contiene los datos de cada ronda en un torneo.		
Atributo	**Tipo**	**Descripción**
Id_Torneo	int	Llave primaria que contiene el identificador de cada torneo
Id_Grupo	int	Llave primaria que contiene el identificador de cada grupo en el torneo.
Id_Ronda	int	Llave primaria que contiene el identificador de cada ronda en cada grupo del torneo.
Fecha	datetime	Campo que contiene la fecha de cada ronda perteneciente al torneo.
HoraInicio	time	Campo contiene la hora en que comienza la ronda en el torneo.

Tabla19. Descripción de la Tabla Partida

Nombre: partida		
Descripción: Esta tabla contiene los datos de las partidas en un torneo.		
Atributo	**Tipo**	**Descripción**
Id_Torneo	int	Llave primaria que contiene el identificador de cada torneo

Id_Grupo	int	Llave primaria que contiene el identificador de cada grupo en el torneo.
Id_Ronda	int	Llave primaria que contiene el identificador de cada ronda en cada grupo del torneo.
Id_Partida	int	Llave primaria que contiene el identificador la partida en cada ronda del torneo.
Id_Jugador1	int	Llave que contiene el identificador del jugador que juega con Blancas en una partida.
Id_Jugador2	int	Llave que contiene el identificador del jugador que juega con Negras en una partida.
Resultado_J_1	enum	Resultado del jugador que juega con Blancas en una partida.
Resultado_J_2	enum	Resultado del jugador que juega con Negras una partida.

Tabla 20. Descripción de la tabla Jugador

Nombre: Jugador		
Descripción: Esta tabla contiene los datos de cada jugador del torneo.		
Atributo	**Tipo**	**Descripción**
Identificador	int	Llave primaria que contiene el identificador de cada jugador
Nombre	varchar	Contiene el nombre de cada jugador
Apellidos	varchar	Contiene los apellidos de cada jugador
Sexo	enum	Contiene el sexo de cada jugador
Rating Elo	int	Contiene el Rating Elo de cada jugador.
K	int	Constante de Experiencia que tiene cada jugador.

Tabla 21. Descripción de la tabla sistemas_desempate

Nombre: sistemas_desempate		
Descripción: Esta tabla contiene los datos de los sistemas de desempates que pueden ser aplicados en un torneo para dar los lugares del mismo. Se suministra con los datos preestablecidos con la Instalación del Sistema, y es de Solo Lectura		
Atributo	**Tipo**	**Descripción**
Identificador	char	Llave primaria que contiene el identificador del sistema de desempate
Nombre	varchar	Contiene el nombre de cada sistema de desempate.

Tabla 22. Descripción de la tabla desempates_aplicados.

Nombre: desempates_aplicados		
Descripción: Esta tabla contiene los desempates aplicados en un torneo.		
Atributo	**Tipo**	**Descripción**
Id_Torneo	int	Llave primaria que contiene el identificador del torneo
Id_Grupo	int	Llave primaria que contiene el identificador de cada grupo en el torneo.
Prioridad	int	Contiene la prioridad en que deben ser usados los sistemas de desempates
Id_desempate	char	Llave que contiene el identificador de cada tipo de desempate aplicado.

Tabla 23. Descripción de la tabla arbitro.

Nombre: arbitro		
Descripción: Esta tabla contiene los datos de los Arbitros.		
Atributo	**Tipo**	**Descripción**
Identificador	int	Llave primaria que contiene el identificador de cada árbitro
Nombre	varchar	Contiene el nombre de cada arbitro
Apellidos	varchar	Contiene los apellidos de cada árbitro
Id_título	varchar	Llave que contiene el identificador del título de cada arbitro .

3.4 Diagramas de Secuencia por Casos de Uso

3.4.1 Diagrama de Secuencia para el Caso de Uso Calcular Rating Elo.

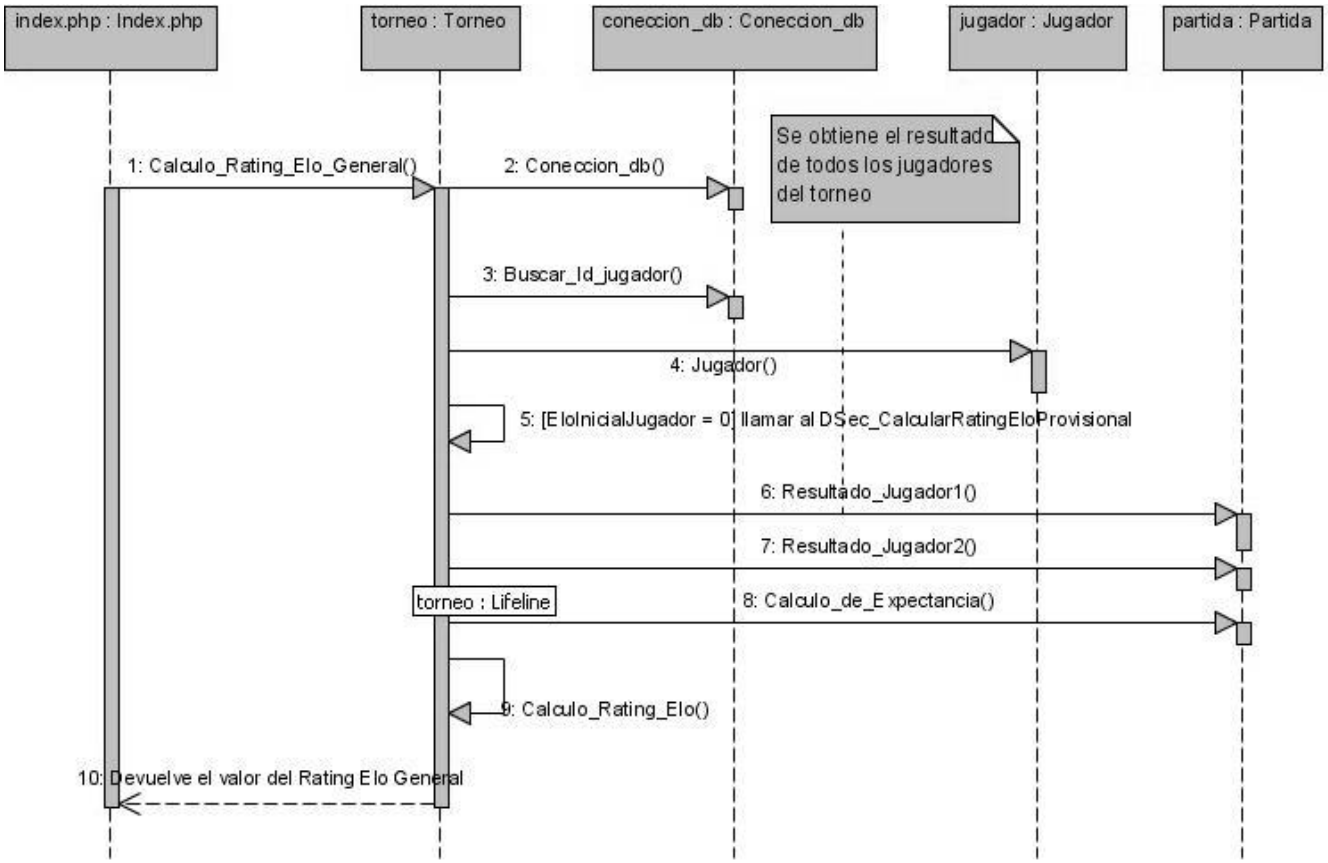


Figura 3.4 Diagrama de Secuencia: Cálculo Rating Elo

3.4.2 Diagrama de Secuencia para el Caso de Uso Procesar Reportes

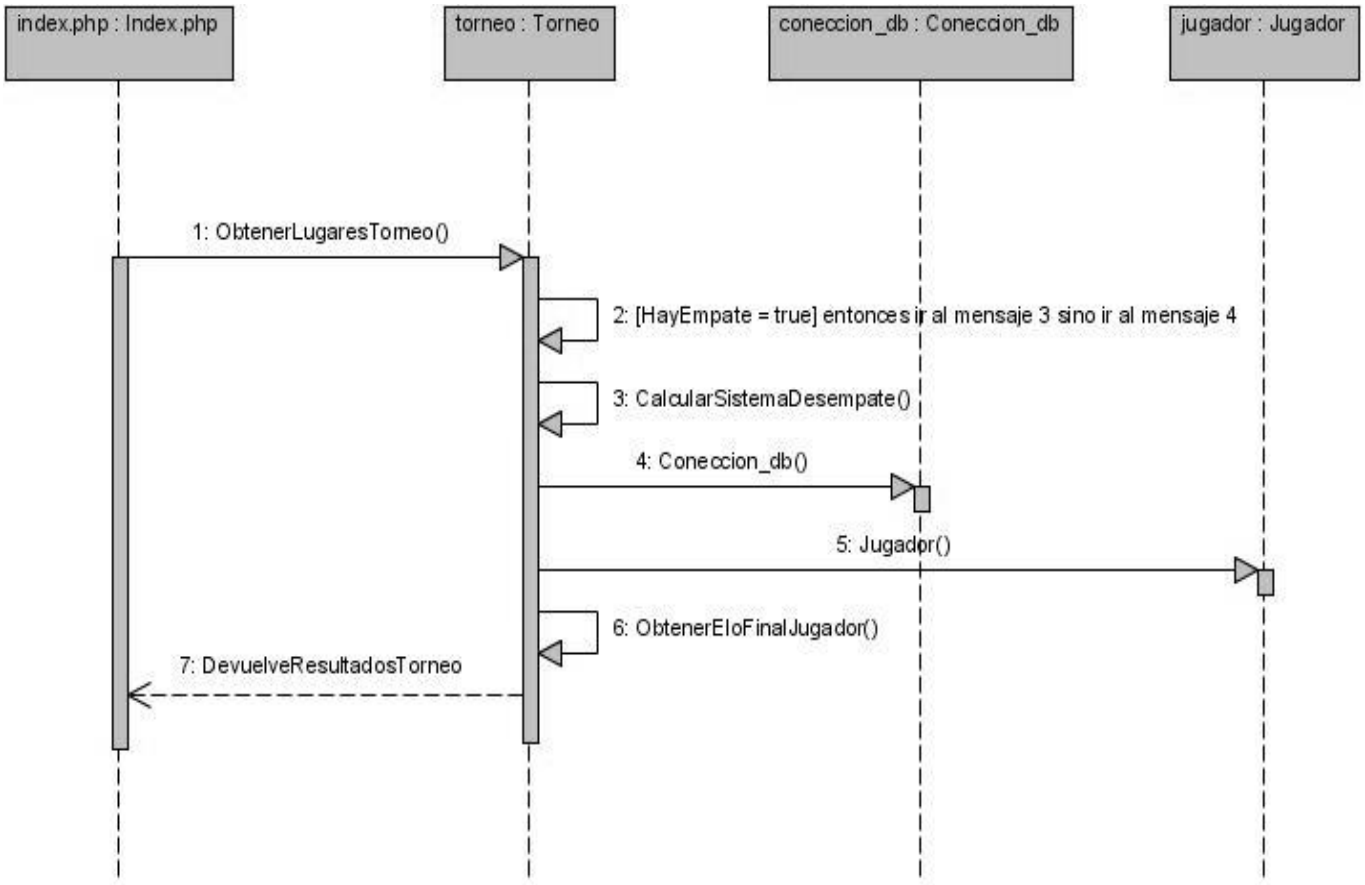


Figura 3.5 Diagrama de Secuencia CU Procesar_Reportes

3.5 Tratamiento de errores

El tratamiento de errores es un factor a tener en cuenta a la hora de desarrollar cualquier aplicación. El sistema propuesto debe ser capaz de manejar los errores que se presenten; de forma tal que afecten en la menor medida posible al usuario. Para evitar la entrada de errores por parte del usuario se establecen tantas validaciones del lado del cliente como resultan posibles, de manera que no lleguen al servidor peticiones incorrectas.

3.6 Seguridad

El tema de crear aplicaciones Web seguras es un tanto complejo, ya que requiere realizar un estudio para comprender los puntos vulnerables de la seguridad. Para la

implementación del sistema propuesto se tuvo en cuenta que la información que se maneja es para el conocimiento de las personas interesadas dentro del área de la cátedra de ajedrez y los estudiantes asociados a ella. Por lo cual desde el primer momento se requerirá acceso a cualquier persona que intente hacer uso del sistema verificándose en primer lugar que sea usuario del dominio UCI.

3.7 Interfaz

La interfaz es la “cara” del sistema, lo cual tiene una gran importancia a la hora de hacer un diseño de la misma. Aunque los elementos de interfaz de este componente son mínimos, para lograr un entendimiento por parte del usuario se creó una interfaz sencilla de manera que el trabajo sea posible de forma intuitiva, sin grandes conocimientos de cómo funciona el sistema. Otro aspecto que se tuvo en cuenta es la organización de la información de forma tal que los datos que se generen tengan el mismo orden dentro de todas las páginas. La uniformidad es otro aspecto que se manejó pues cada elemento estará regido por el mismo estilo de tamaño, forma y color dentro del sitio.

3.8 Análisis de costo y beneficios

Dado el uso extendido y su probada eficacia, la metodología RUP fue la escogida para este trabajo. Esta metodología esta basada en Casos de Uso lo cual representa una ventaja a la hora de hacer un levantamiento correcto de los requerimientos del sistema. Los requerimientos quedan recogidos en términos de actores y casos de uso. Los actores representan, como se ha visto a lo largo del presente trabajo, tanto a un usuario humano como a otros sistemas informáticos que interactúan con el sistema en cuestión. Los Casos de Uso son funcionalidades que le brindan valor a un actor determinado. Al ser las relaciones entre actores y casos de uso una forma de obtener los requerimientos pueden dar una medida de la cantidad de trabajo necesario para llevar a cabo el proyecto y por tanto su costo. La estimación mediante el análisis de Puntos de Casos de Uso es un método propuesto originalmente por Gustav Karner de Objectory AB, y posteriormente refinado por muchos otros autores. Se trata de 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. Para el cálculo del esfuerzo requerido de este proyecto será utilizado el método anteriormente expuesto.

Puntos de Casos de Uso

El primer paso es el cálculo de puntos de casos de uso sin ajustar. Lo cual se hace a partir de la siguiente ecuación:

UUCP = UAW + UUCW

Donde:

UUCP: Puntos de casos de uso sin ajustar.

UAW: Factor de peso de los actores sin ajustar.

UUCW: Factor de peso de los casos de uso sin ajustar.

Cálculo de **UAW**

Tabla 24. Tipos de actores y sus pesos.

Actores del sistema:	Tipo de actor	Factor de Peso
Reloj	Complejo	3
Jugador	Complejo	3

UAW= Sumatoria de la multiplicación de la cantidad de actores de un tipo específico con su factor de peso.

Cantidad de actores de tipo complejo: 2

UAW = 2 x 3

UAW =6.

Cálculo de **UUCW**

Éste se obtiene a partir de los casos de uso del sistema y su complejidad. La complejidad está dada por la cantidad de secuencias atómicas que tenga el caso de uso.

Tabla 25. Tipos de Casos de Uso y sus pesos.

Casos de uso	**Tipo de Caso de Uso**	**Factor de Peso**

Mostrar reporte por cada Ajedrecista en torneo...	Simple	5
Calcular Variación Rating Elo	Simple	5
Calcular Rating Provisional de un ajedrecista.	Simple	5
Calcular Expectancia	Simple	5
Obtener Puntos Alcanzados	Simple	5
Obtener Datos	Simple	5

UUCW = Sumatoria de la multiplicación de la cantidad de Casos de Uso de un tipo específico con su factor de peso.

Cantidad de casos de uso simples: 6

UUCW = 6 x 5

UUCW = 30

Al ser **UAW = 6** y **UUCW = 30** y **UUCP = UAW + UUCW** sustituyendo tenemos:

UUCP = UAW + UUCW

UUCP = 6 + 30

UUCP = 36

Después de haber calculado los puntos de Casos de Uso sin ajustar se calculan los puntos de Casos de Uso ajustados.

UCP = UUCP x TCF x 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.

Cálculo del 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.

Tabla 26. Factor de Complejidad Técnica.

Factor	**Descripción**	**Peso**	**Valor asignado**
T1	Sistema distribuido	2	1
T2	Objetivos de performance o tiempo de respuesta	1	1
T3	Eficiencia del usuario final	1	1
T4	Procesamiento interno complejo	1	1
T5	El código debe ser reutilizable	1	0
T6	Facilidad de instalación	0.5	2
T7	Facilidad de uso	0.5	2
T8	Portabilidad	2	2
T9	Facilidad de cambio	1	3
T10	Concurrencia	1	1
T11	Incluye objetivos especiales de seguridad	1	1
T12	Provee acceso directo a terceras partes	1	1
T13	Se requieren facilidades especiales de entrenamiento	1	0

El Factor de complejidad técnica se calcula mediante la siguiente ecuación:

TCF = 0.6 + 0.01 x Σ (Pesoi x Valor asignadoi)

TCF=0.6+0.01x Σ(2x1+1x1+1x1+1x1+1x0+0.5x2+0.5x2+2x2+1x4+1x1+1x1+1x1+1x0)

TCF=0.6 + 0.01 x 17

TCF=0.6+0.17

TCF=0.78

Cálculo del Factor Ambiente (**EF**)

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos actores son los que se contemplan en el cálculo del Factor de ambiente.

Tabla 27. Factor Ambiente.

Factor	**Descripción**	**Peso**	**Valor asignado**
E1	Familiaridad con el modelo de proyecto utilizado	1.5	1
E2	Experiencia en la aplicación	0.5	1
E3	Experiencia en trabajo orientado a objetos	1	5
E4	Capacidad del analista líder	0.5	2
E5	Motivación	1	5
E6	Estabilidad de los requerimientos	2	5
E7	Personal a tiempo completo	-1	0
E8	Dificultad del lenguaje de programación	-1	0

El Factor de ambiente se calcula mediante la siguiente ecuación:

EF =1.4 - 0.03 x Σ (Peso x Valor asignado)

EF =1.4 – 0.03 x Σ (1,5x3+0.5x3+1x5+0.5x3+1x5+2x5+(-1x0)+(-1x0))

EF = 1.4 – 0.03 x 27.5

EF = 0.575

Al ser **UUCP = 36**, **TCF=0.77**, **EF = 0.575** y **UCP = UUCP x TCF x EF** entonces:

UCP=44 x 0.78 x 0.575

UCP=15,939

El esfuerzo en horas-hombre viene dado por:

E = UCP x CF

Donde:

E= Esfuerzo

UCP= Puntos de Casos de Uso ajustados (calculado anteriormente).

CF= Factor de conversión (para este tipo de proyecto 20 horas-hombre/Punto de Casos de Uso)

Al tener **UCP = 15,939**, **CF = 20 horas-hombre/Punto de Casos de Uso** y **E = UCP x CF**

E=15,939 x 20 horas-hombre/Punto de Casos de Uso

E=318,78 horas-hombre

Tabla 28. Esfuerzo por flujo de trabajo.

Actividad	Porcentaje	Esfuerzo
Análisis	10%	79,7
Diseño	20%	159,4
Implementación	40%	318,78
Prueba	15%	119.55
Sobrecarga	15%	119.55
Total	100%	797

El proyecto requerirá de 797 horas-hombre a desarrollar por una persona en un tiempo de 133 días; tomando que se trabajará en el proyecto como promedio 6 horas diarias.

Por lo cual se puede decir que se terminará el proyecto en aproximadamente 4 meses ajustándose a los requerimientos del cliente.

3.8.1 Beneficios

El presente trabajo, como parte de las aplicaciones que se desarrollan con el objetivo de ser usadas dentro de la Universidad de las Ciencias Informáticas, no reporta un beneficio monetario directo. Sin embargo; desde el punto de vista de los beneficios intangibles que reporta, se puede decir que con el subsistema de Cálculo de la Variación del Rating Elo y Reportes, se logrará volver a establecer el Rating Elo UCI, que alguna vez debió suspenderse por la imposibilidad de llevarlo manualmente. Esto incentivará mucho la práctica del Ajedrez de Alto Rendimiento en nuestra Universidad, lo que redundará en provecho de nuestros estudiantes y trabajadores.

Conclusiones

Con el desarrollo de este capítulo se logró una comprensión del problema que permitió la implementación del sistema propuesto. Se modeló el diagrama de clases del análisis que brindó una visión de lo que debe hacer el sistema. Con el diagrama de clases del diseño se logró determinar las clases que serán implementadas así como sus atributos y sus responsabilidades. Además se realizó un análisis de costo beneficio el cual demostró la factibilidad de llevar a cabo el proyecto.

CAPÍTULO 4. IMPLEMENTACIÓN

4. Introducción

En este capítulo se expondrán el diagrama de despliegue y el diagrama de implementación. Estos diagramas son muy importantes dentro del flujo de trabajo de implementación porque representan la distribución del sistema propuesto. Se describirán además los componentes que son necesarios para que la aplicación propuesta funcione. El objetivo principal de esta disciplina es convertir los elementos del diseño en elementos de implementación, dichos elementos son códigos fuentes, ejecutables, binarios, entre otros.

4.1 Implementación

4.1.1 Diagrama de despliegue

El diagrama de despliegue representa la distribución física de la aplicación. La estructura de nodos que se utiliza en éste, está diseñada para permitir que el usuario tenga comodidad en el trabajo además de propiciar que exista una seguridad confiable. Se utiliza un servidor Web que tendrá la aplicación en sí; en otro nodo se encontrará la Base de Datos del sistema y por último están las estaciones terminales que son las computadoras de la UCI desde donde se podrá acceder al sistema para utilizar los servicios que presta. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución, como un computador, un dispositivo o memoria. Los estereotipos permiten precisar la naturaleza del equipo:

- Dispositivos
- Procesadores
- Memoria

Figura 4.1 Diagrama de Despliegue

Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, bibliotecas cargadas dinámicamente, etc. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.

El diagrama de componente forma parte de la vista física de un sistema, la cual modela la estructura de implementación de la aplicación por sí misma, su organización en componentes y su despliegue en nodos de ejecución. Esta vista proporciona la oportunidad de establecer correspondencias entre las clases y los componentes de implementación y nodos. La vista de implementación se representa con los diagramas de componentes.

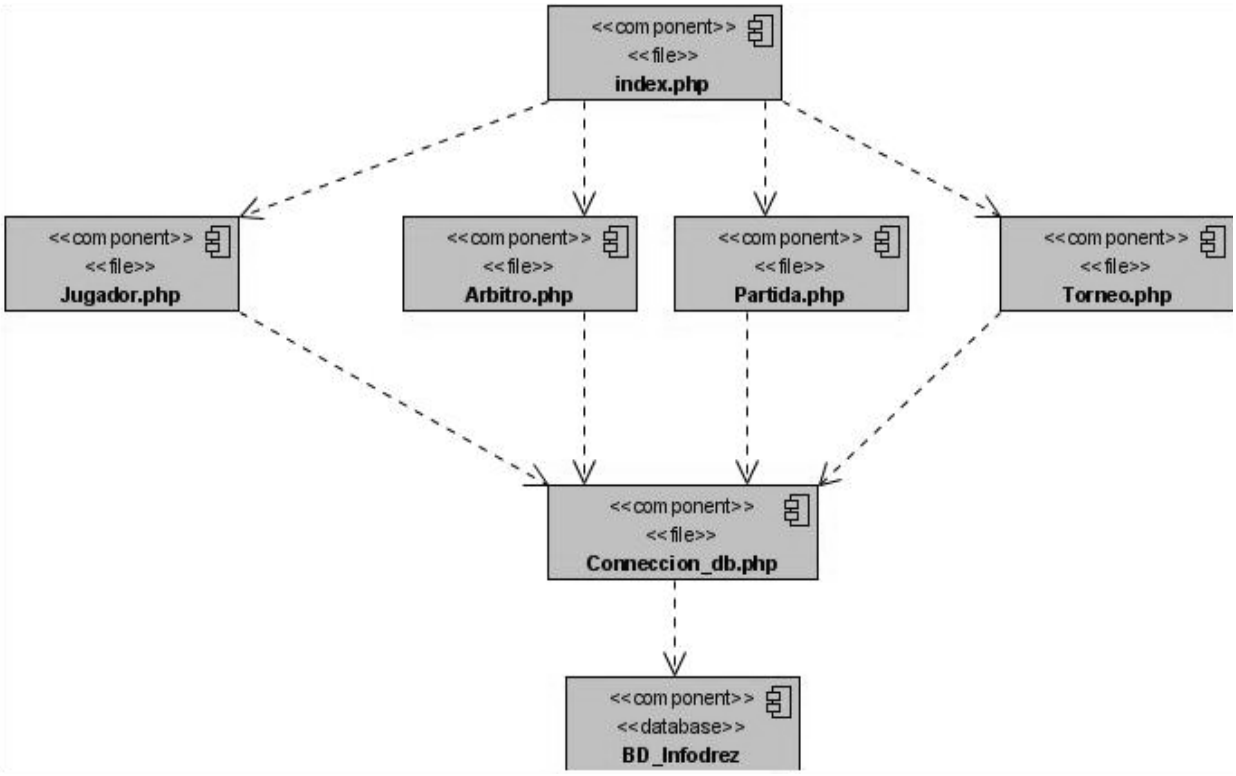


Figura 4.2 Diagrama de Componentes.

Conclusiones

En este capítulo se representaron los componentes a ser implementados así como el diagrama de despliegue. Se logró obtener un mapa de la implementación del sistema propuesto; el cual sirvió de guía para lograr un software que responda a los requerimientos del cliente.

CONCLUSIONES

- Se realizó un estudio del entorno de trabajo, que permitió comprender los procesos que se ejecutan por un Arbitro asociados a las funciones de Cálculo de la Variación de Rating Elo.
- Se describieron los procesos, lo cual brindó una mayor claridad a la hora de implementar el sistema.
- Se modelaron conceptualmente las clases que están implicadas en el sistema.
- Se desarrollaron los diagramas de actividades, lo cual ayudo a comprender los procesos de negocio.
- Se desarrolló una aplicación, integrada al componente de Arbitraje del Proyecto Infodrez, que será instalada en la Cátedra de Ajedrez de la UCI, y permitirá volver a activar el Rating Elo UCI para los Torneos Individuales en modalidad Round Robin.
- Se integraron los conocimientos adquiridos en el transcurso de la carrera en las disciplinas de Ingeniería de Software y Programación.

RECOMENDACIONES

- Continuar el estudio de los procesos asociados al Cálculo del Rating Elo, para Torneos Individuales en modalidades: Suizos y Muerte Súbita, así como para Torneos por Equipos, con vistas a realizar la Ingeniería de Software e Implementarlos, reutilizando el código que ya se posee, e incorporándolo como parte de este producto.

- Integrar este componente al resto de la plataforma Infodrez, en su evolución como sistema de gestión de contenidos (CMS).

REFERENCIAS BIBLIOGRÁFICAS

1. **WEEKS, MARK.** Introduction to Chess Ratings. [En línea] 2007. http://chess.about.com/mbiopage.htm.

2. Introduction to Chess Ratings. [En línea] 2007. [Citado el: 7 de mayo de 2008.] http://chess.about.com/mbiopage.htm.

3. **MICHELONE, MANUEL LÓPEZ.** El rating de los jugadores de ajedrez. [En línea] [Citado el: 7 de mayo de 2008.] http://www.chessbase.com/espanola/newsdetail2.asp?id=3078.

4. **REYNOX.** Metodología de Desarrollo de Software (MDS). [En línea] 2007. http://www.reynox.com.ar/sistemas/metodologia.php#link2

5. **SONAS, JEFF.** Rating Formula. [En línea] 2002. [Citado el: 14 de mayo de 2008.] http://www.chessbase.com/newsdetail.asp?newsid=562

6. **FENACH.** Federación Nacional de Ajedrez en Chile. [En línea] 2008. [Citado el: 21 de mayo de 2008.] http://www.fenach.cl/

7. **ACHOUR, MEHDI.** Manual de PHP. [En línea] 25 de febrero de 2006. [Citado el: 25 de mayo de 2008.] http://www.php.net/manual/es/introduction.php.

8. **Apexnet.** Comparación de Herramientas de modelado UML Enterprise Architect y Rational Rose. [En línea] 6 de junio de 2005. [Citado el: 12 de enero de 2008.] http://www.apexnet.com.ar/index.php/news/main/38/event=view.

9. Ajedrez por Aguas Calientes. [En línea] 1 de Abril de 2008. [Citado el: 1 de junio de 2008.] http://ajedrezxaguascalientes.blogspot.com/2008/04/elo-fide-abril-2008.html.

10. **RUMBAUGH, JAMES.** El lenguaje unificado de modelado. [En línea] [Citado el: 11 de enero de 2008.] http://bibliodoc.uci.cu/pdf/reg03050.pdf

11. Estilos y Patrones. [En línea] 2007. [Citado el: 1 de junio de 2008.] http://siona.udea.edu.co/~aoviedo/Arquitectura%20de%20Software/EstilosyPatrones.htm.

12. **ROSS, DANIEL.** Arpad Elo y sistema de puntuación Elo Otoño . [En línea] 2007. [Citado el: 12 de enero de 2008.] http://www.chessbase.com/espanola/newsdetail2.asp?id=5639.

13. **Alfonso, Manuel.** Cálculo del Elo. [En línea] 2007. [Citado el: 11 de enero de 2008.] http://members.fortunecity.es/unichessclub/calculoelo.htm.

Bibliografía

EGF. 2006. Sistema de puntuación. *Como Calcular tus puntos ¿?* [En línea] 2006. [Citado el: 19 de marzo de 2008.] http://www.clubgomadrid.org/articles/ptosegf.php.

A.I. Gerardo Anaya Rodríguez, AN Bárbara López Chávez - Meriño. ISLA. [En línea] [Citado el: 3 de marzo de 2008.] http://www.inder.cu/capablanca/Anaya/Anaya01.HTM.

2006. Ajedrez en Colombia. [En línea] 2006. [Citado el: 5 de marzo de 2008.] http://www.ajedrezencolombia.com/Arpad_Elo.html.

Ajedrez Menorca. [En línea] Inter Ajedrez. [Citado el: 10 de junio de 2008.] http://www.oliware.com/ajedrez/calc_elo.htm.

Ajedrez por Aguas Calientes. [En línea] [Citado el: 7 de febrero de 2008.] http://ajedrezxaguascalientes.blogspot.com/2008/04/elo-fide-abril-2008.html.

1997_2003. Ajedrez Siglo 21. [En línea] 1997_2003. [Citado el: 15 de febrro de 2008.] http://www.ajedrezsiglo21.com/Rumores/pedidoafide.asp.

Alfonso, Manuel. Cálculo del Elo. [En línea] [Citado el: 1 de febrero de 2008.] http://members.fortunecity.es/unichessclub/calculoelo.htm.

ALICANTE. Club de Ajedrez ALICANTE. [En línea] [Citado el: 3 de febrero de 2008.] http://www.ajedrezalicante.com/calculo_del_elo.htm.

2008. Articulo de Enciclopedia. [En línea] 2008. [Citado el: 16 de marzo de 2008.] http://enciclopedia.us.es/index.php/Elo.

Arturo Vaccaro, Esteban Girón. 1998. Federación Cubana de Ajedrez. *Francisco Acosta Ruiz.* [En línea] Argentina, Panamá, 1998. [Citado el: 25 de marzo de 2008.] http://www.cuba.cu/ajedrez/boletin/editorial23.html.

ASOCIACION ESPAÑOLA DE AJEDREZ POR CORRESPONDENCIA. [En línea] [Citado el: 27 de enero de 2008.] http://www.matepostal.com/Documentos/Rating%20Elo.htm.

Barroso, José Luis Rodríguez. FISE. [En línea] Venezuela. [Citado el: 25 de marzo de 2008.] http://www.scrabble.grupos.usb.ve/downloads/EL%20ESCALAFON%20ELO.pdf.

2007. Club Ajedrez VICAR. [En línea] 2007. [Citado el: 7 de marzo de 2008.] http://www.clubajedrezvicar.com/modules.php?name=News&file=article&sid=210.

Club Aranjuez de Ajedrez. [En línea] http://www.ajedrezaranjuez.com/elo.htm.

EL PORTAL DE AJEDREZ. [En línea] [Citado el: 5 de febrero de 2008.] http://www.schackportalen.nu/Espanol/esspelare.htm.

2002. elo en Gennio. [En línea] 2002. [Citado el: 9 de marzo de 2008.] http://www.gennio.com/tags/elo/populares/3.

2006. ELO FEDA. [En línea] 2006. [Citado el: 14 de marzo de 2008.] http://www.feda.org/circulares_detall.php?id=36&circular=Circular%2003/2008.

2007. Enrocate. [En línea] 2007. [Citado el: 5 de febrero de 2008.] http://www.enrocate.com/modules.php?name=News&file=article&sid=899.

FAREP-FENAP. [En línea] [Citado el: 6 de marzo de 2008.] http://www.ajedrezpanama.org/elo/elo.htm.

FENACH. [En línea] [Citado el: 10 de febrero de 2008.] http://www.fenach.cl/modules.php?name=News&file=article&sid=120.

González, Romelio Milián. 2008. CAPABLANCA. [En línea] 2008. [Citado el: 6 de abril de 2008.] http://www.inder.cu/capablanca/Milian/Milian01.HTM.

2008. IndalChess. [En línea] 2008. [Citado el: 13 de marzo de 2008.] http://www.indalchess.com/noticias/index.php?option=com_content&task=view&id=15&Itemid=39.

Inter Ajedrez. [En línea] [Citado el: 10 de enero de 2008.] http://www.interajedrez.com/elo/elo.htm.

LEBREDO, GERARDO. Bohemia_Revista de Análisis General fundada en 1908. *Frente a la élite.* [En línea] [Citado el: 3 de abril de 2008.] http://www.bohemia.cu/2006/10/12/unpoco/ajedrez.html.

Michelone, Manuel López. 2004. chessbase. [En línea] junio de 2004. [Citado el: 2008 de febrero de 2008.] http://www.chessbase.com/espanola/newsdetail2.asp?id=3078.

2007. NORMATIVA GENERAL DE COMPETICIONES OFICIALES...FEDERACIÓN NAVARRA DE AJEDREZ. [En línea] SEPTIEMBRE de 2007. [Citado el: 26 de marzo de 2008.] http://www.clubajedrezorvina.com/uploads/Normativa2007-2008.pdf.

Oswaldo Ceballos Burbano, Jose Orlando Ruiz. 2006. Resolución # 099. [En línea] Jasbon, 2006. [Citado el: 27 de marzo de 2008.] http://www.torneosajedrez.com/ajedrez47/eventos/181/resolucion_elo_nacional.html.

2006. Pasión por el Ajedrez. [En línea] 2006. [Citado el: 20 de marzo de 2008.] http://pasionporelajedrez.blogspot.com/2006/06/como-se-calcula-el-puntaje-elo-primera.html.

2006. Rating Programas. [En línea] 2006. [Citado el: 1 de abril de 2008.] http://www.schackportalen.nu/Espanol/esrating.htm.

Ross, Daniel. 2007. Arpad ELo y su sistema de puntuación. [En línea] 2007. [Citado el: 4 de abril de 2008.] http://nau64.blogspot.com/2007/12/arpad-elo-y-el-sistema-de-puntuacin-elo.html.

Ruiz, Ing. Jose Orlando. Jasbon. [En línea] [Citado el: 25 de febrero de 2008.] http://www.torneosajedrez.com/ajedrez47/eventos/265/Normas_en_Finales_Nacionales.html.

Sánchez, Fermín Fernández. 2007. Valoracion del Elo. [En línea] 2007. [Citado el: 15 de febrero de 2008.] http://www.geotour.es/ELO/valoracion_elo.pdf.

2007. SIDURCHESS. [En línea] 2007. [Citado el: 4 de marzo de 2008.] http://sidurchess.com/modules.php?name=News&file=print&sid=353.

2004_2005. SPANISH_ARENA.com. [En línea] 2004_2005. [Citado el: 18 de marzo de 2008.] http://www.spanish-arena.com/inicio/index.php?op=1000&sop=19.
Wikipedia. [En línea] [Citado el: 1 de marzo de 2008.] http://es.wikipedia.org/wiki/Elo.

Zaiat, Alfredo. 2006. Diario. [En línea] 2006. [Citado el: 23 de marzo de 2008.] http://www.pagina12.com.ar/diario/economia/2-68157-2006-06-10.html.

GLOSARIO DE TÉRMINOS

CASE: (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

HTML: (HyperText Markup Language, lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

HTTP: (*HyperText Transfer Protocol,* protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la Web. El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido.

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

MVC: Modelo Vista Controlador es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

OMG: (Object Management Group) es una corporación independiente y sin ánimo de lucro. Tiene el propósito de crear una arquitectura estándar para objetos distribuidos en redes (componentes).

UML: (*Unified Modeling Language*, Lenguaje Unificado de Modelado). Es un lenguaje de modelado de sistemas de software.

ANEXOS

Para jugadores avanzados, se han establecido una serie de clasificaciones que en parte se basan en si un jugador es capaz de obtener una serie de resultados mínimos en ciertos torneos, contra jugadores de un nivel predeterminado, y en parte, y cada vez más comúnmente, en su fuerza evaluada de acuerdo con todos sus resultados y resumida en la puntuación Elo : El sistema de puntuación Elo es un método para calcular la fuerza relativa de los jugadores de juegos como el ajedrez. Fue inventado para mejorar el sistema de clasificación vigente de los jugadores de ajedrez. A menudo se escribe ELO (con letras mayúsculas) en la errónea creencia de que se trata de un acrónimo o para distinguir al sistema de su inventor, el profesor Arpad Elo (1903-1992), un físico estadounidense de origen húngaro.

Elo FIDE: Se dice que un jugador de ajedrez tiene categoría FIDE cuando consigue un porcentaje de puntuación en sus enfrentamientos con contrincantes que previamente ya han obtenido dicha categoría.

Para determinar cuando un jugador está incluido en la lista FIDE se recurre a las normas establecidas por dicha federación. Se deben contabilizar 9 partidas disputadas con jugadores que ya lo hayan conseguido, establecer la media de éstos y conseguir un porcentaje de puntos tal que alcance los 1401.

Se puede conseguir este logro en un solo torneo o bien en varios, siempre con bloques iguales o superiores a tres enfrentamientos, y sumando al menos un punto, teniendo en cuenta que dichos bloques caducan si no se alcanza en un tiempo determinado el resto de partidas para conformar el bloque de nueve partidas.

La lista FIDE incluye jugadores cuyo rating sea superior a los 1401 puntos.

El Elo FIDE existen otras categorías inferiores basadas en el mismo sistema. Así podemos distinguir un Elo nacional, Elo autonómico ó Elo provincial.