UNIVERSIDAD DE LAS CIENCIAS INFORMATICA

FACULTAD 5

“Entornos Virtuales”

Juego de Dominó para Jugadores Virtuales

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

Autor: Roberto Linares Rubio

Tutor: Lic. José Miguel de la Rosa.

Ciudad de la Habana

“Año 50 de la Revolución”

"No necesito saberlo todo. Tan sólo necesito saber dónde encontrar lo que me haga falta, cuando lo necesite..."

(Albert Einstein).

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Firma del Autor	Firma del Tutor
(Roberto Linares Rubio)	(Lic. José Miguel de la Rosa)

Datos de Contacto

Lic. José Miguel de la Rosa Trevín.

Graduado de Ciencias de la Computación en el 2006 en la Universidad de la Habana. Profesor del Departamento de Programación desde septiembre de 2006. Profesor Adiestrado. Actualmente cursando una Maestría en Ciencias de la Computación en la Universidad de la Habana. Perteneciente al proyecto Uvi-bot y jefe de la línea de Jugadores Virtuales.

Agradecimientos

Gracias a la vida que me ha dado tanto.

A mis padres Mayda y Rosendo, a quienes les debo la vida y viviré eternamente agradecido por inculcarme siempre sus sabios consejos y experiencias.

A mi hermano Rosendito (Chicho) por darme la oportunidad de servirle de guía y ejemplo.

A mi novia Meylin por el apoyo, la confianza, preocupación, comprensión y por estar siempre ahí en los momentos más difíciles de mi carrera.

A todos mis compañeros de año por darme su apoyo y ayuda, en especial a Alberto Pacheco (el Pache), que me ayudó incondicionalmente en los momentos más difíciles de esta etapa.

A las personas que de una forma u otras aportaron su granito de arena para que lograra hacer este sueño realidad y nunca dejaron de confiar en mí.

Dedicatoria

A mis padres,
mi hermano y mi novia...

Resumen

La mayoría de las universidades están incorporando los juegos virtuales a su sistema de enseñanza, con el objetivo de poner a prueba la capacidad de sus alumnos en la resolución de las situaciones más complejas y cambiantes del mercado internacional.

En la Universidad de las Ciencias Informáticas se han realizado algunos juegos con el objetivo de que los estudiantes tengan una alternativa diferente y amena para la práctica de las técnicas de la programación aprendidas, pues en ocasiones la enseñanza actual se torna un poco pasiva y al incorporar un juego al proceso educativo, el alumno sale de esa pasividad.

El presente trabajo brinda una propuesta de un juego virtual del popular jugo de dominó, el cual brinda la oportunidad a los estudiantes de que programen sus propias estrategias del juego, demostrando así el dominio de los diferentes lenguajes de programación aprendido. Las actividades lúdicas pueden ayudar a motivar al alumno y retenerlo hasta que complete sus estudios.

Palabras Claves

Juegos Virtuales: Los denominados juegos para jugadores virtuales se basan en darle al usuario la posibilidad de crear su jugador virtual brindándole una serie de funcionalidades y ejemplos que lo ayuden y faciliten su creación. Estos juegos generalmente se dividen en dos aplicaciones que unidas conforman el juego.

Jugador Virtual: Un jugador virtual es un personaje inmerso en un mundo virtual, creado por un usuario, el cual elige las características que tendrá su jugador virtual, así como sus funcionalidades, por ejemplo: su raza, profesión, armas, estrategias, etc. e ir incrementado su nivel de complejidad en todas sus etapas.

Lúdica: Perteneciente o relativo al juego.

Tabla de Contenido

Introducción

Los juegos de multimedia son una de las mayores atracciones de los niños, los adolescentes y los jóvenes de la última década. La explicación de este fenómeno, se basa principalmente, en que estamos frente a una nueva generación, los denominados nativos digitales.

Sus características e intereses, los llevan a querer ser partícipes necesario y sobre todo protagonistas de sus propias aventuras, mas que meros observadores pasivos de las historias creadas por otros. Este escenario, pone al sistema educativo y al mundo laboral, ante el dilema de incorporar las nuevas tecnologías y en especial los videojuegos al aula, al trabajo y al aprendizaje; enfrentarse a él o acompañar a la nueva generación de los nativos digitales, en la exploración, el análisis crítico, la comprensión y el buen uso de estas herramientas, con el fin de potenciar su proceso de aprendizaje.

El juego es una de las más potentes estrategias de simulación que existe. Usualmente se considera que los juegos digitales perjudican la inteligencia del jugador en lugar de desarrollarla. Sin embargo, se está tornando más claro, que los juegos digitales han realizado una combinación exitosa de estrategias complejas, significados persuasivos para sostener la atención y la inteligencia del jugador. En los juegos digitales, el participante, tiene la posibilidad de realizar acciones y ver los resultados de sus decisiones, esto más allá de un sentido trivial, redefine la palabra "jugar".

El desarrollo de habilidades, información y conocimiento se torna más accesible por fuera de los confines de la educación y entrenamiento formal. Los juegos comerciales orientados al entretenimiento continúan avanzando a pasos agigantados en su habilidad de atraer, sostener y mantener a los jugadores atrapados e inmersos en un juego placentero por propia voluntad.

Diversos estudios han demostrado el potencial del juego para el aprendizaje, sin embargo se conocen solo algunas experiencias concretas que utilicen el juego virtual y las TICS con fines educativos. Las características particulares de los entornos virtuales colaborativos, parecen ser el canal propicio para integrar el juego en el aprendizaje y el trabajo.

El juego nos permite convertirnos en verdaderos protagonistas de nuestro proceso de aprendizaje, la tecnología nos ofrece velocidad, interactividad y la posibilidad de colaborar e interactuar con otros en tiempo real, a través de diversas herramientas y canales. Por lo tanto es tiempo de jugar y tenemos que preparar nuestros jóvenes para enfrentar su futuro, no nuestro pasado".

Situación Actual

La Universidad de las Ciencias Informáticas (UCI) es una institución de nueva creación ya que es la primera Universidad creada al calor de la Batalla de Ideas, a la cual se le ha asignado debido a su importancia, una serie de proyectos productivos tanto de alcance nacional como internacional.

Para darle cumplimiento a estos proyectos los estudiantes deben dominar las técnicas de programación que se imparten en la Universidad. Desde sus inicios se ha podido observar que los estudiantes no presentan suficiente interés por su aprendizaje, pues al principio, debido a la forma en que se impartía la asignatura, los estudiantes de una forma u otra se fueron alejando de su práctica diaria, pues sentían que su enseñanza era un poco pasiva, todo el tiempo era frente a un profesor o a una tele clase, en otras ocasiones planteaban que no encontraban un motivo suficiente para acercarse a su estudio, en las mayorías de las ocasiones los estudiantes empleaban su tiempo en la práctica de juegos virtuales donde se pudo observar un gran interés y motivación por los mismos.

En la Universidad se estuvo incursionando en la creación de algunos juegos, donde el usuario se convertiría en un jugador virtual y tendría que demostrar toda su destreza y habilidad, a través de todos sus conocimientos acerca de la programación. Se pudo comprobar que hubo una gran aceptación por parte de los estudiantes de este tipo de juego y a la vez se comprobó de que la universidad no contaba con suficientes aplicaciones de este tipo, lo cual se dio la tarea de incursionar en la realización de nuevas opciones, esta vez en la realización del popular juego de dominó, planteándose el siguiente problema:

Problema a resolver:

¿Cómo implementar un juego de dominó para Jugadores Virtuales?

La decisión al escoger este tipo de juego, es sencilla pues superado únicamente por el béisbol como deporte nacional, no hay dudas de que el juego de dominó es uno de los pasatiempos más difundidos por todo el archipiélago cubano.

El presente trabajo de diploma se plantea como **objetivo general** desarrollar un juego de dominó para jugadores virtuales, donde el usuario tenga la posibilidad de implementar el desempeño de su jugador virtual.

Como **objeto de estudio** se plantea los juegos para jugadores virtuales y **campo de acción** los juego de dominó para jugadores virtuales.

Para logar el objetivo planteado se proponen las **tareas** que se mencionan a continuación

- Estudiar las técnicas y reglas que rigen el juego de dominó.
- Instruirse acerca de todas las posibles estrategias que se pueden aplicar en el juego.
- Aprender el trabajo con las herramientas que se puedan utilizar para la modelación e implementación del Juego.
- Asimilar las ventajas y desventajas que pueda traer consigo el tipo de aplicación y el lenguaje de programación.
- Realizar un estudio minucioso de juegos similares en los que se apliquen técnicas iguales, principalmente en la UCI.
- Implementar un conjunto de funcionalidades que constituyan las herramientas básicas en la implementación de los jugadores.
- Definir la terminología y conceptos a utilizar en el desarrollo del juego de dominó.

Aportes prácticos esperados del trabajo.

Si se logra hacer un juego de Dominó para jugadores Virtuales, donde los estudiantes, profesores y todo aquel interesado, pueda vincular las técnicas de programación con la practica de este popular juego, lograríamos contar con una opción más amena y atractiva donde los usuarios vinculen sus conocimientos con las actividades lúdicas.

Este trabajo está estructurado en 4 capítulos, distribuidos de la siguiente forma:

- **Capítulo 1. Fundamentación Teórica**: Breve reseña del estado del arte, definición de que algunos conceptos importantes, descripción de las características de algunos juegos similares existente actualmente en el mundo. Tendencias y Tecnologías Actuales: Descripción de tendencias y tecnologías seleccionadas para el desarrollo de la propuesta de solución.

- **Capítulo 2. Características del Sistema**: Modelo de Dominio; requisitos funcionales y no funcionales; actores y casos de uso del sistema.

- **Capítulo 3. Análisis y Diseño del Sistema**: Descripción del análisis y diseño a través de diagramas de clases del análisis y diseño de la aplicación.

- **Capítulo 4. Implementación**: Descripción de los modelos de implementación, los diagramas de despliegue y de componentes.

Capítulo 1. Fundamentación teórica.

1.1 Introducción

Un jugador virtual es un personaje inmerso en un mundo virtual, creado por un usuario, el cual elige las características que tendrá su jugador virtual, así como sus funcionalidades, por ejemplo: su raza, profesión, armas, estrategias, etc. e ir incrementado su nivel de complejidad en todas sus etapas.

En la mayoría de los casos nos referimos a un jugador virtual cuando hablamos de un jugador programado por el usuario, el cual antes de comenzar el juego debe indicarle la estrategia a usar, y una serie de reglas en las cuales se basa el personaje para tomar decisiones y jugar lo más racional posible una vez introducido en el mundo virtual del juego. La inteligencia de dicho jugador dependerá de la habilidad de su creador al programarlo.

En el capítulo se muestra un estudio sobre las definiciones de juegos para jugadores virtuales, además se hace referencia al estado del arte de los juegos virtuales, su antecedentes y tendencias actuales.

Finalmente se realizó un estudio de las diferentes herramientas y lenguajes de modelación para la construcción del juego de dominó para jugadores virtuales.

1.2 Juegos para jugadores virtuales. Definiciones.

Los denominados juegos para jugadores virtuales se basan en darle al usuario la posibilidad de crear su jugador virtual brindándole una serie de funcionalidades y ejemplos que lo ayuden y faciliten su creación. Estos juegos generalmente se dividen en dos aplicaciones que unidas conforman el juego. La primera aplicación es la que permitirá al usuario la creación de su jugador virtual y la segunda parte seria la interfaz del juego como tal que se dedicaría a cargar y enfrentar los jugadores virtuales programados por el usuario con anterioridad.

La aplicación de estos juegos en la educación es muy ventajosa, ya que permite motivar el interés del usuario por la programación; el juego logrará con su capacidad de atraer nuestra atención y generar un profundo interés por la actividad, mantenernos inmerso en ese mundo virtual del juego artificial y el usuario ansioso por ganar y acercarse a la meta cada vez un poco más, llegado a este punto el juego logra alcanzar su meta, logrando motivar al usuario a programar ,ya que es la única forma de crear un buen jugador virtual.

De esta forma el juego para jugadores virtuales contribuye a desarrollar la lógica de razonamiento del usuario como su habilidad de programar. O sea, eliminamos el tabú que algunas personas plantean de que el juego solo es pérdida de tiempo convirtiéndose en un eficaz método de práctica.

1.3 Juegos Virtuales de Dominó

A la hora de desarrollar este trabajo fue necesario, el estudio de diferentes tipos de juego de dominó, al comenzar la búsqueda me percaté de que la gran mayoría de los juegos que existen de este tipo ninguno aplica la técnica de crear un jugador virtual, son juegos para jugar online. Por ejemplo:

Jamaican Dominoes

Juego de dominó online en flash, no es un juego de dominó multijugador pero podrás jugar al dominó solo con otros jugadores virtuales.

Real Dominoes 1.0

Real Dominoes es un juego que hará las delicias tanto de los expertos jugadores de dominó como de los principiantes. Te garantiza diversión y aprendizaje a partes iguales, independientemente de tu nivel de habilidad. El juego esta construido bajo un motor de dominó muy potente que te permitirá ir mejorando tu juego poco a poco.

Aprendes a jugar, inventa o perfecciona nuevas estrategias, compite contra jugadores de todo el mundo. El juego tiene un estupendo entorno gráfico 3D, y te permite jugar contra otro rivales controlados por el ordenador.

xFX Domino 8.0.5

Se trata de unos de los juegos de mesa más populares y extendidos que existen. Para aquellos que no hayan jugados nunca es bien sencillo: con las fichas que dispones, las cuales tienen dos valores (uno por cada lado de la ficha) debes ir colocándolas en la mesa al lado de otra ficha que tenga algunos de esos dos valores, el que antes se deshaga de todas las fichas será el ganador.

Al realizar un estudio de la aceptación de estos tipos de juego se comprobó que los usuarios poco a poco se iban desmotivando de este tipo de juego, pues en ocasiones se tornaba pasivo, pues ellos mismos ya querían imponer su propia estrategia de juego, convirtiéndose así en jugadores virtuales para poder llevar a cabo su inteligencia y creatividad. Al comprobarse todo esto se llegó a la conclusión, de la necesidad de estos tipos de juegos en la universidad.

1.3 Estado del Arte

La posibilidad de utilizar un ordenador para jugar en redes telemáticas comenzó en torno a 1979, cuando un grupo de estudiantes de la Universidad de Essex crearon una versión informática multiusuario de un juego de rol llamado Dragones y Mazmorras, basado en el uso de textos alfanuméricos. Así surgió un nuevo tipo de juegos conocidos como MUD (Multi-User Dungeons o Domains) que se desarrollaría rápidamente por la aún poco conocida Internet, surgiendo así las primeras comunidades virtuales.

El primer juego multiusuario que incorporó imágenes fue Hábitat en 1986, creado por Lucas Films Games y destinado para el Commodore 64. De él surgieron posteriormente juegos como el EverQuest, Asheron's o Ultima Online.

Pero la verdadera revolución de los juegos en red surgió en 1993 con la creación de la World Wide Web. Los usuarios tenían la posibilidad de acceder gratuitamente a versiones reducidas de videojuegos para ordenador con fines básicamente promocionales. Además la rápida difusión de Internet como medio de entretenimiento facilitó la mejora de las tecnologías para la conexión en red de usuarios, siendo en la actualidad World of Warcraft el juego más destacable online.

Es importante destacar también el auge de las videoconsolas que, desde principios de la década de los 90, ofrecen la posibilidad del juego online. La primera consola que incorporó la posibilidad de conexión a Internet para jugar en red fue Dreamcast, que se lanzó en Japón en 1998. A partir de ella las consolas más importantes empezaron a ofrecer la posibilidad de ser conectadas a Internet.

1.3.1 Antecedentes

En la actualidad el mundo de la Informática se desarrolla con gran rapidez, lo cual ha permitido que todo lo referente al mundo real se haya llevado a un plano virtual. La importancia de la informática en la animación cinematográfica y televisiva es cada vez mayor. Pero no nos engañemos, el ámbito donde las creaciones gráficas computarizadas tienen una mayor relevancia es el mundo de los juegos de ordenador. De hecho, un programa de juegos no es otra cosa que una gama de animaciones digitales de manejo interactivo. Por otro lado, la importancia comercial de este sector es tan grande que incluso ha dado lugar a un amplio abanico de productos derivados. Todo ello convierte a la historia del videojuego en un relato fascinante que vale la pena conocer.

El mundo actual tiene gran inclinación sobre los videojuegos, debido a que los juegos virtuales se han convertido en uno de los principales mercados de la rama. Los juegos para jugadores virtuales han influido en su auge aunque en un por ciento discreto por ser una línea más novedosa dentro de este mundo. Los usuarios con el desarrollo de los juegos exigen cada vez más interactividad con el juego, deseando adaptar al máximo los jugadores virtuales a sus preferencias. Existen diferentes aplicaciones que permiten al usuario alcanzar gran parte del protagonismo en el desempeño del jugador virtual que lo representa, ejemplo de juegos de jugadores virtuales a nivel internacional son: SimpleSoccer, Javasoccer, Hundirla flota.

SimpleSoccer

Es un demo que viene documentada su implementación y objetivo en uno de los libros más famosos del mundo actual, “Programming Game AI by Example”, cuyo objetivo es iniciar a cualquier usuario interesado en el mundo de la programación de los videojuegos, tenga un alto nivel de las técnicas y lógica de programación o no, y explicar como se define su estructura e implementa un juego sencillo de fútbol en este caso.

La aplicación consiste en simular un partido de fútbol entre dos equipos virtuales cuyas tácticas por ser un demo netamente educativo vienen ya implementadas. La aplicación esta constituida por

varios componentes, por ejemplo: el físico que se encarga de simular los fenómenos físicos del juego (principalmente la realidad de la visualización del movimiento de los jugadores virtuales y la pelota) y el componente de IA que se encarga de la toma de dediciones a nivel de equipo, entre otros. Una vez que comienza el partido el usuario no podrá intervenir en el mismo a no ser como simple espectador.

Aquí faltaría adicionar la posibilidad al usuario de crear su equipo e implementar su táctica antes de cargarlo y comenzar así el partido. Por ser un demo educativo no lo posee.

Estos juegos motivan al usuario a tratar de mejorar sus habilidades en la programación debido a que están fuertemente motivados por el resultado del juego que depende de la eficacia de su implementación y su correcta selección de elementos como formación, estrategia entre otros aspectos en la confección de su equipo virtual.

Javasoccer

Es una aplicación que simula un partido de fútbol aplicada desde hace cinco años en la Universidad de Girona con el objetivo de fomentar la comparación e integración de las técnicas de IA. Concretamente en las prácticas, los alumnos deben desarrollar un equipo de jugadores de fútbol que decidan las acciones a realizar basándose en varias de las técnicas de IA. Al final del curso, se realiza una competición entre los diferentes equipos, donde los estudiantes pueden evaluar y comparar objetivamente los resultados. La motivación que adquieren al competir entre ellos en partidos reales es clave para despertar su interés hacia la IA.

HundirLaFlota

Es un juego para jugadores virtuales de acción (guerra) que se esta utilizando en un concurso en Internet. El juego consiste en una batalla naval en el que cada jugador posee una flota de barcos a la cual debe posicionar estratégicamente en el campo de batalla (tablero), programar su estrategia de combate, así como un conjunto de características entre las que resalta el nombre de la flota; gana el primero que hunda la flota del contrario. Como juego para jugadores virtuales el usuario debe programar todas estas características y comportamientos en la confección de la flota antes de comenzar la batalla, una vez comenzada el jugador no puede intervenir.

En Cuba, a pesar de que es una rama novedosa en la que se ha trabajado poco, se ha desarrollado y puesto en práctica una aplicación de este tipo en la UCI denominada **BattleProgUci**, específicamente en la facultad 5. El juego consiste en una pelea entre dos o más jugadores virtuales en un mundo artificial que posee además vidas, municiones y obstáculos. Ésta aplicación es un juego para jugadores virtuales de acción (guerra) que brinda la posibilidad al usuario de adaptar al jugador que lo representa a su preferencia, ya que en sus manos queda la implementación de las principales características y comportamientos del jugador que concluirán en el buen o mal desempeño del mismo dentro del mundo artificial que constituye el juego. En fin, dotar de cierta independencia al jugador de su creador una vez comenzada la pelea.

1.3.2 Mundo Virtual

Los juegos se hicieron para ser jugados por más de una persona, desde los clásicos de mesa, como el dominó y el ajedrez, hasta sofisticados juegos de acción en primera persona. Con el desarrollo de Internet, se potenciaron los entretenimientos en red, donde lo único que cambia es el soporte, en este caso digital.

El nuevo medio permitió, también, la emergencia de mundos virtuales con posibilidades de interacción impracticables en el mundo real. Surgieron así los llamados Juegos de Rol Multijugador Masivos en línea donde el participante se compenetra con un personaje que vive una vida paralela. Argentum Online, Tribal Wars, Lineage II: The Chaotic Chronicle, Guild Wars, World of Warcraft y Second Life son algunos ejemplos significativos de estos nuevos espacios.

Mediante los juegos se pueden aprender conceptos, estrategias cognitivas para la resolución de problemas, desarrollar la atención, la memoria, la fluidez verbal y numérica, el razonamiento, las capacidades creativas y de expresión gráfica o musical, son instrumentos que pueden usarse en

diferentes momentos del proceso de aprendizaje: previamente como elemento motivador, simultáneamente al desarrollo de los contenidos como complemento, o posteriormente en forma de refuerzo o profundización. Siempre habrá que adecuar el juego a la edad y necesidades del usuario en función de los objetivos que se quieran conseguir.

1.3.2 Tendencias Actuales

Los juegos virtuales actuales se acercan a lo real, pero no lo suficiente. Los creadores de estos juegos tienen claro hacia donde deben dirigir todos sus esfuerzos, hacia la hiperrealidad. El desarrollo de tecnologías de captación del movimiento puede permitir que nuestros representantes en juegos virtuales nos imiten hasta tal punto que acabemos creyendo que no son personajes de ficción, sino actores reales. La finalidad es alcanzar una emotividad radical derivada del entretenimiento.

Los juegos hiperrealistas, que no nos permitirán establecer la diferencia entre un personaje de ficción y otro real, son el futuro de los juegos virtuales, cuyo desarrollo depende ahora más del avance de los dispositivos de interacción que del diseño de los juegos en sí.

Las tecnologías de captación del movimiento, con las que los ordenadores analizan el movimiento de los jugadores, que van equipados con sensores, y los reproducen, han avanzado mucho en los últimos tiempos, pudiendo llegar en el futuro a reproducir expresiones faciales. La clave radica en el desarrollo de métodos de captación del movimiento (incluso el de las expresiones faciales, los movimientos de los ojos, cada pequeña arruga que se forme o cada inclinación de la cabeza). Así, el personaje que represente al jugador no podrá ser más real.

Hasta ahora, los creadores de esta modalidad de ocio tecnológico no han sido capaces de desarrollar personajes con expresiones y rostros tan realistas como los de las fotografías o el cine, pero todo puede cambiar. Y se calcula que en el plazo de tan sólo dos años. La fórmula es aplicar características humanas a los personajes para que resulten completamente convincentes. Ése es el próximo paso que deben dar los programadores de los juegos.

El desarrollo de tecnologías de captación del movimiento puede permitir que nuestros representantes en juegos virtuales nos imiten hasta tal punto que acabemos creyendo que no son personajes de ficción, sino actores reales. El final será alcanzar una emotividad radical derivada del entretenimiento. Otro paso importante será que los juegos sean cada vez más adaptables a los

jugadores, de tal forma que observándoles aprendan y se modifiquen a la medida de los usuarios. La nueva generación de juegos de ficción casi reales no será sólo lúdica, sino que también servirán como entrenamiento emocional para personas que deban enfrentarse posteriormente a situaciones extremas.

1.4 Herramientas

A la hora de desarrollar este juego, primeramente fue preciso el estudio minucioso de las posibles herramientas que me servirían para su construcción, inclinándome finalmente por el entorno de desarrollo Microsoft Visual Studio .Net 2005.

1.4.1 Microsoft Visual Studio .Net 2005

Visual Studio .NET es un conjunto de aplicaciones completo para la creación tanto de aplicaciones de escritorio como de aplicaciones Web para el trabajo en equipo. Aparte de generar aplicaciones de escritorio de alto rendimiento, se pueden utilizar las eficaces herramientas de desarrollo basado en componentes y otras tecnologías de Visual Studio para simplificar el diseño, desarrollo e implementación en equipo de soluciones.

Características

Con Visual Studio 2005, los desarrolladores profesionales pueden:

- **Disfrutar un entorno de desarrollo altamente productivo:** con diseñadores visuales, lenguajes de programación y editores de código mejorados.
- **Desarrolla y depura aplicaciones multicapa de servidor:** desde un mismo entorno unificado de desarrollo (Integrated Development Enviroment - IDE).
- **Construye soluciones para SQL Server 2005:** utilizando herramientas visuales integradas de diseño de bases de datos e informes.
- **Crea tus propias herramientas que extienden el IDE de Visual Studio:** usando el SDK de Visual Studio.
- **Crear aplicaciones de línea de negocio:** usando Visual Basic, C#, C++ y J#
- **Construir aplicaciones para Windows, la Web y dispositivos móviles:** todo desde el mismo entorno unificado de desarrollo.
- **Desarrollar aplicaciones cliente/servidor:** usando servicios Web y herramientas integradas de diseño para acceder a datos remotos.

1.4.2 Lenguaje de programación.

C y C++ son dos de los lenguajes más utilizados en el campo de la ingeniería y la programación de sistemas. La principal razón es que C y C++ proporcionan el nivel de abstracción preciso para construir una aplicación compleja, pero, al mismo tiempo, ofrecen mecanismos de bajo nivel que permiten a los programadores hacer uso de las características más avanzadas de las plataformas sobre las que se ejecutan sus programas. Microsoft ha creado C# que combina algunas de las características más avanzadas de Java con algunas de las más potentes de C y C++. La idea es convertirlo en el nuevo lenguaje de Internet y, por supuesto, en el lenguaje nativo para acceder a todos los servicios que en el futuro brindará .NET.

Por estas razones y por las que a continuación se muestran, se basó mi elección por este lenguaje.

1.4.2.1 Lenguaje de programación C#

C# es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg, éste último también conocido por haber sido el diseñador del lenguaje Turbo Pascal y la herramienta RAD Delphi.

Es actualmente uno de los lenguajes de programación más populares en la informática y las comunicaciones. El objetivo de Microsoft, es permitir a los programadores abordar el desarrollo de aplicaciones complejas con facilidad y rapidez. Es como si tomáramos todas las cosas buenas de Visual Basic y las añadiéramos a C#, aunque recortando algunas de las tradiciones más ocultas y difíciles de conocer de C y C++. Con C# no sólo se pueden escribir programas para la Web, sino que también permite desarrollar aplicaciones de propósito general.

Características

Con la idea de que los programadores más experimentados puedan obtener una visión general del lenguaje, y el porque de mi elección les presento de manera resumida algunas de las principales características de C#.

• **Sencillez:** C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET, el código escrito en C# es auto contenido, lo que significa que no necesita de ficheros adicionales a la propia fuente, tales como ficheros de cabecera o ficheros IDL y el tamaño de los tipos

de datos básicos, es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile, lo que facilita la portabilidad del código y no se incluyen elementos pocos útiles.

• **Modernidad:** C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose que son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, la inclusión de una instrucción **foreach** que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico **string** para representar cadenas o la distinción de un tipo **bool** específico para representar valores lógicos.

• **Orientación a objetos:** Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos. C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo. En lo referente a la encapsulación es importante señalar que aparte de los típicos modificadores **public**, **private** y **protected**, C# añade un cuarto modificador llamado **internal**, que puede combinarse con protected e indica que al elemento a cuya definición precede, sólo puede accederse desde su mismo ensamblado.

• **Orientación a componentes:** La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).

• **Gestión automática de memoria:** Como ya se comentó, todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje, de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se active –ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente-, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción **using**.

• **Eficiente:** En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador **unsafe**) y podrán usarse en ellas punteros de forma similar a cómo se

hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grandes.

• **Compatible:** Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java, que permite incluir directamente en código escrito en C#, fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados Platform Invocation Services (PInvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32. Nótese que la capacidad de usar punteros en código inseguro, permite que se pueda acceder con facilidad a este tipo de funciones, ya que éstas muchas veces esperan recibir o devuelven punteros. También es posible acceder desde código escrito en C# a objetos COM. Para facilitar esto, el .NET Framework SDK incluye una herramientas llamadas **tlbimp** y **regasm** mediante las que es posible generar automáticamente clases proxy que permitan, respectivamente, usar objetos COM desde .NET como si de objetos .NET se tratase y registrar objetos .NET para su uso desde COM. Finalmente, también se da la posibilidad de usar controles ActiveX desde código .NET y viceversa. Para lo primero se utiliza la utilidad **aximp**, mientras que para lo segundo se usa la ya mencionada **regasm**.

1.4.2.2 Decisión de lenguaje a utilizar.

Para desarrollar la propuesta que presenta este trabajo, se ha decidido utilizar después de un estudio de los diferentes lenguajes de programación C#, porque es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la VCL usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el .NET Framework SDK.

1.5 Lenguaje de Modelación

Para modelar el análisis y el diseño del software se utilizará el lenguaje UML que es el más utilizado a nivel mundial en la actualidad. UML (Lenguaje Unificado de Modelación), es el lenguaje de modelado visual que se utilizará para especificar, visualizar, construir y documentar los artefactos del sistema. De esta manera poder entender, diseñar, configurar, mantener y controlar la información del sistema a construir.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar

un trabajo que finalmente beneficia a un usuario externo. Con este lenguaje de modelado se pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.

1.5.1 ¿Por qué se utilizará Rational Unified Process? (RUP)

Se utilizará RUP por capturar varias de las mejores prácticas en el desarrollo moderno de software en una forma que es aplicable para un amplio rango de proyectos y organizaciones. Servirá de guía para utilizar de manera efectiva UML y le proporcionará a cada miembro de un equipo un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas de desarrollo. Creará y mantendrá modelos, en lugar de enfocarse en la producción de una gran cantidad de papeles de documentación.

Características de RUP:

Iterativo e Incremental: El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio sólo consta de varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

Dirigido por casos de uso: En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc.

Centrado en la arquitectura: El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios del mismo: electricidad, fontanería, etc. Así mismo ocurre con el desarrollo de un sistema software.

Enfocado en los riesgos: El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración, deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

1.5.2 ¿Por qué se utilizará Rational Rose?

Se utilizará Rational Rose por ser una buena solución de modelado visual en el mundo, y una excelente herramienta para traducir requisitos de alto nivel a una arquitectura flexible basada en componentes. Rational Rose por encontrarse a la cabeza en cuanto al desarrollo del Unified Modeling Language (UML), se ha convertido en la notación estandarizada empleada en Rational Rose para especificar, visualizar y construir desarrollos de software y sistemas. Rational Rose domina el mercado de herramientas para el análisis, modelamiento, diseño y construcción orientada a objetos.

Rose es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros del equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las grandes ventajas de Rose es que utiliza la notación estándar en la arquitectura de software (UML), la cual permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común, además los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto.

Capítulo 2. Características del sistema

2.1 Introducción

En el siguiente capítulo se analiza la definición de los procesos, situación problémica, objeto de automatización, Modelo de Dominio, actores, trabajadores, así como su descripción, selección de los requisitos funcionales y no funcionales y modelo de Casos de Uso del Sistema.

2.2 Situación Problémica

La Universidad de Ciencias Informáticas (UCI) por sus características especiales la hacen un objetivo estratégico para el desarrollo de las ciencias informáticas en el país, a la misma se le han asignado una serie de proyectos de colaboración de trabajo, con empresas nacionales e internacionales. Para darle cumplimiento a dichos compromisos es necesario que los estudiantes dominen las técnicas de programación. La Universidad de Ciencias Informáticas (UCI) necesita tomar medidas y encontrar posibles soluciones para lograr que los estudiantes se vinculen con la práctica de esta asignatura.

A pesar de que nuestra institución cuenta con una amplia base de conocimiento y bibliográfica y un universo de profesores bien calificado, esto no es suficiente. Los estudiantes plantean que su aprendizaje es muy monótono y pasivo, lo cual contribuye a que se alejen de su práctica.

2.3 Objeto de Automatización

La Facultad 5 ha tomado la decisión de implementar un sistema en el cual los estudiantes encuentren en el mismo una forma amena y diferente para la práctica de las técnicas de programación. En este caso se desarrollará un sistema que consiste en la implementación del popular juego de dominó, donde dará la posibilidad de que los estudiantes se conviertan en jugadores virtuales.

2.4 Sistema Propuesto

El sistema se dividirá en dos partes: la parte visual o gráfica le dará la posibilidad al usuario de jugar una partida, podrá seleccionar la cantidad de jugadores (dll) que desee que participen en la partida (dos, tres o cuatro), además el usuario tendrá la posibilidad de configurar su propia partida, o sea, le dará un tiempo permisible de aceptación de la dll y escogerá el tipo de juego que quisiera desarrollar entre jugadores (uno-uno, tres (de forma individual), cuatro (de forma individual) o en

pareja), podrá ordenar a simulará toda la partida y posteriormente tendrá la posibilidad de salvar dicha partida para disponer de ella cuando estime conveniente.

La otra parte del sistema será el resultado de una muestra visual del fin de una partida o del juego, donde el usuario tendrá la oportunidad de ver todo lo ocurrido en la partida acción por acción.

2.5 Modelo de Dominio

Considerando las descripciones del sistema propuesto en el epígrafe anterior, se llega a la conclusión de que el negocio que se está estudiando tiene muy bajo nivel de estructuración, con soluciones muy diversas y dispersas. Por lo que se utilizará un modelo de dominio pues permite mostrar de manera visual los principales conceptos que se manejan en el dominio de la aplicación en desarrollo y de esta forma utilizar un vocabulario común que ayude al jugador a entender el contexto en que se ubica la solución propuesta, logrando una captura correcta de requisitos. (Ver Figura 2.1)

2.5.1 Clases conceptuales

En este epígrafe se identificarán los conceptos utilizados en el modelo del dominio representado en el siguiente diagrama:

- **Juego:** Es el controlador de todo el proceso de simulación del juego de dominó, encargado de chequear reglas y llevar el control de puntuación, pases, turnos, etc.

- **Jugador:** Se le denominará a la entidad encargada de controlar los datos de un jugador y de llevar a cabo sus acciones.

- **DLL Interface:** Es la entidad que garantiza la comunicación con la Dll, ya sea de tipo estándar o de framework.

- **Historial:** Es el conjunto de todas las acciones realizadas dentro de las partidas de un juego.

- **Controlador:** Es la entidad que controla el comportamiento de la Dll que juega, en caso de mal funcionamiento de la Dll o bucles infinitos, la interrumpe y devuelve resultados.

2.5.2 Diagrama de clases del Modelo de Dominio.

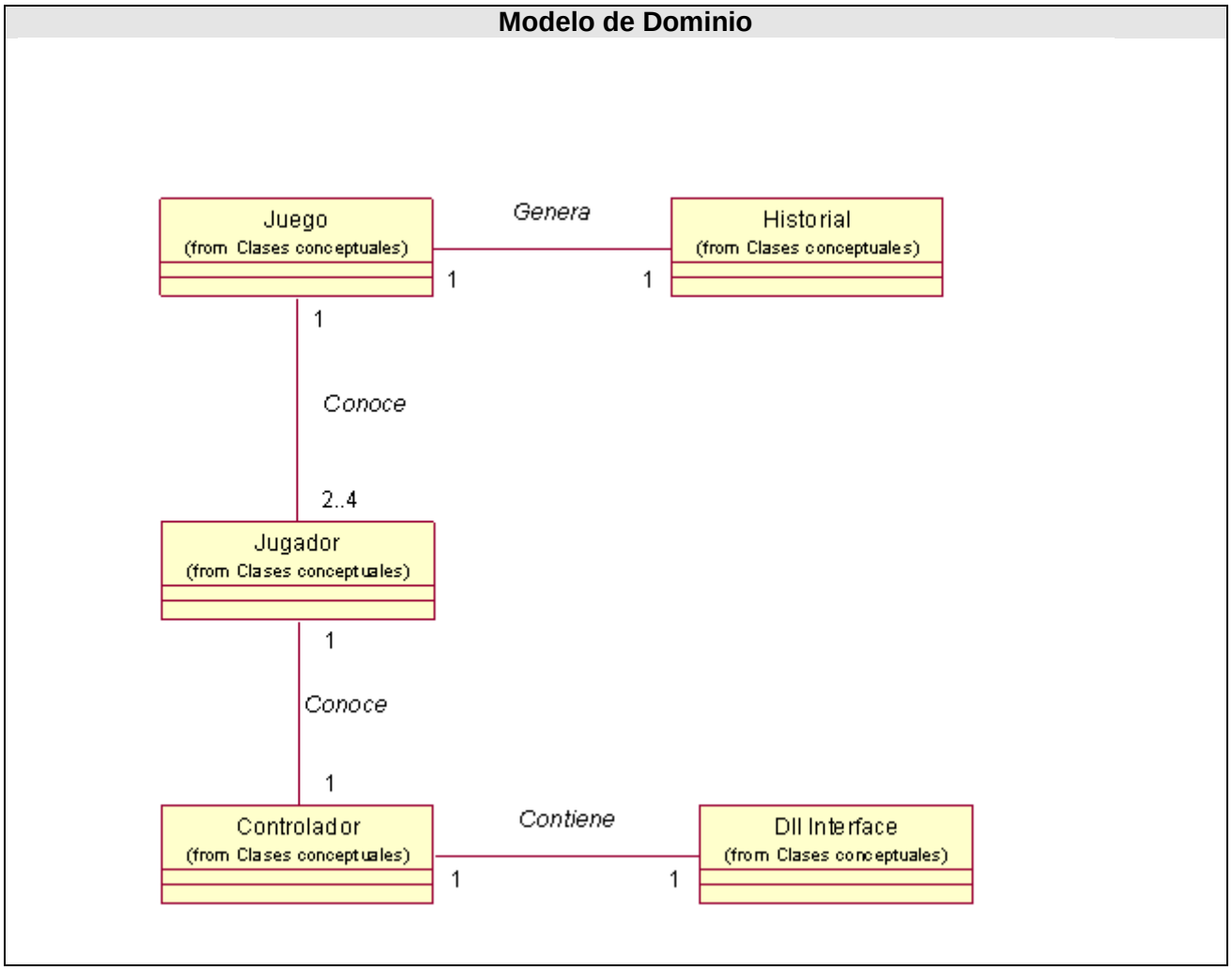

Figura 2.1 Diagrama de clases del Modelo de Dominio.

2.6 Levantamiento de requisitos

Lograr una comunicación efectiva entre los jugadores y el desarrollador del proyecto con el objetivos de llegar a un entendimiento de lo que hay que hay que hacer, es la clave del éxito en la producción de un software. El propósito fundamental del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto.

2.6.1 Requerimientos Funcionales del Sistema.

Los requerimientos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física. Es por ello que su definición debe ser clara y libre de ambigüedades.

A continuación se muestran los requerimientos funcionales definidos para la realización de la aplicación a desarrollar. Estos requisitos están fraccionados según un grupo de funciones del sistema las cuales se agruparan para formar los casos de uso del sistema.

R1. Jugar Partida

- Seleccionar jugadores (DLLs)
- Eliminar Jugadores
- Configurar la partida
- Simular

R2. Salvar Partida

R3. Ver Partida

- Mostrar fichas jugadas y fichas de jugadores al estilo natural.
- Mostrar mensajes de acciones a ejecutar por las dlls.
- Recorrer la lista de acciones hacia delante y hacia a tras.
- Ver los resultados de cada ronda.

R4. Cargar Partida

- Seleccionar la partida.

2.6.2 Requerimientos no Funcionales del Sistema.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente están vinculados a requerimientos funcionales, es decir una vez que

se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación, por lo que se consideran fundamentales en el éxito del producto.

A continuación se presentan los requerimientos no funcionales definidos para la realización de la aplicación a desarrollar.

Apariencia o interfaz externa:

- Diseño orientado a llamar la atención del jugador y con una navegación sencilla.
- Construcción de enlaces rápidos.

Usabilidad:

- El Software podrá ser usado por jugadores que posean conocimientos básicos en el manejo de la computadora y de programación.

Rendimiento:

- Tiempo de respuesta rápida.

Portabilidad:

- Debe ser capaz de poder ejecutarse en cualquier maquina mientras este instalado el framework de .NET con muy poca configuración.

Funcionalidad:

- Reducir al mínimo el tiempo en que se carga el juego virtual.

Software:

- Requiere el framework de .NET y los sistemas operativos que soporten dicho framework.

2.7 Determinación y justificación de los actores de sistema.

De acuerdo a lo expuesto en los epígrafes anteriores se determinó que el sistema tendrá que interactuar solamente con el actor usuario. En la tabla que aparece a continuación mostrará la descripción de este actor. (Ver Tabla 2.1)

Actor	Justificación
Usuario	Usuario es cualquier persona que utilice e interactué con el sistema. Estos usuarios pueden ser los estudiantes, los profesores o cualquier otra persona que desee utilizar la aplicación.

Tabla 2.1 Descripción del Actor

2.8 Diagrama de Casos de Uso del Sistema.

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores. (Ver Figura 2.2)

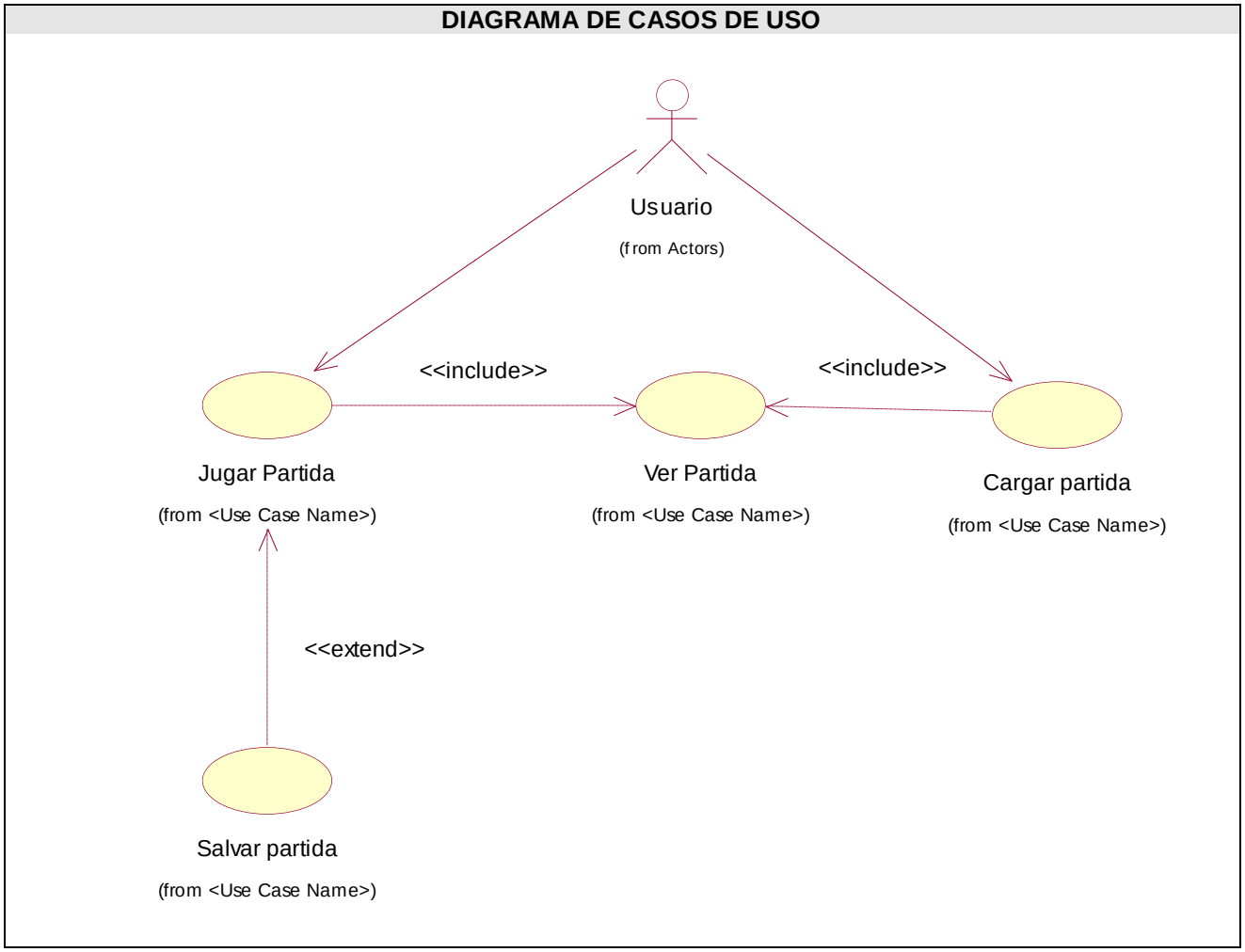

Figura 2.2 Diagrama de Casos de Uso del Sistema

2.8.1 Casos de Uso del Sistema

En este epígrafe se mostrará los casos de uso del sistema con una breve descripción. (Ver Tablas 2.2, 2.3, 2.4 y 2.5)

CU- 1	Jugar Partida.
Actor	Usuario.
Descripción	Dar la posibilidad de jugar una partida.
Referencia	R1

Tabla 2.2 Caso de Uso: Jugar Partida

CU- 2	Salvar Partida.
Actor	Usuario.
Descripción	El usuario tendrá la oportunidad de salvar la partida en el lugar que desee para luego analizarla.
Referencia	R2

Tabla 2.3 Caso de Uso: Salvar Partida

CU- 3	Ver Partida.
Actor	Usuario.
Descripción	El usuario tendrá la posibilidad de ver el resultado de la partida, acción por acción.
Referencia	R3

Tabla 2.4 Caso de Uso: Ver Partida

CU- 4	Cargar Partida.
Actor	Usuario.
Descripción	El usuario tendrá la posibilidad de cargar y ver como se desarrolló la partida acción por acción.
Referencia	R4

Tabla 2.5 Caso de Uso: Cargar Partida

2.8.2 Descripción textual de los Casos de Uso del Sistema

En este epígrafe se realizará una detallada descripción textual de los casos de uso del sistema y se mostrará la pantalla correspondiente. (Ver Tabla 2.6, 2.7, 2.8 y 2.9)

Caso de uso	
CU- 1	Jugar Partida.
Propósito	Jugar una partida.
Actores	Usuario.
Resumen	El caso de uso se inicia cuando el usuario solicita jugar una partida.
Referencias	RF1

Acción del actor	**Respuesta del sistema**
1 El usuario solicita jugar una partida	1.1 El sistema muestra una interfaz en la cual brinda la posibilidad de configurar la partida y seleccionar los jugadores (DLLs)
2 El usuario selecciona los jugadores, elimina algún jugador, configura el tiempo por llamada, el tipo de juego (en caso de seleccionarse los cuatros jugadores, si desea jugar en pareja) y por ultimo simula la partida.	
3. El usuario da la orden de simular.	3.1 El sistema simula el juego y desencadena el caso de uso Ver Partida.

Pantalla

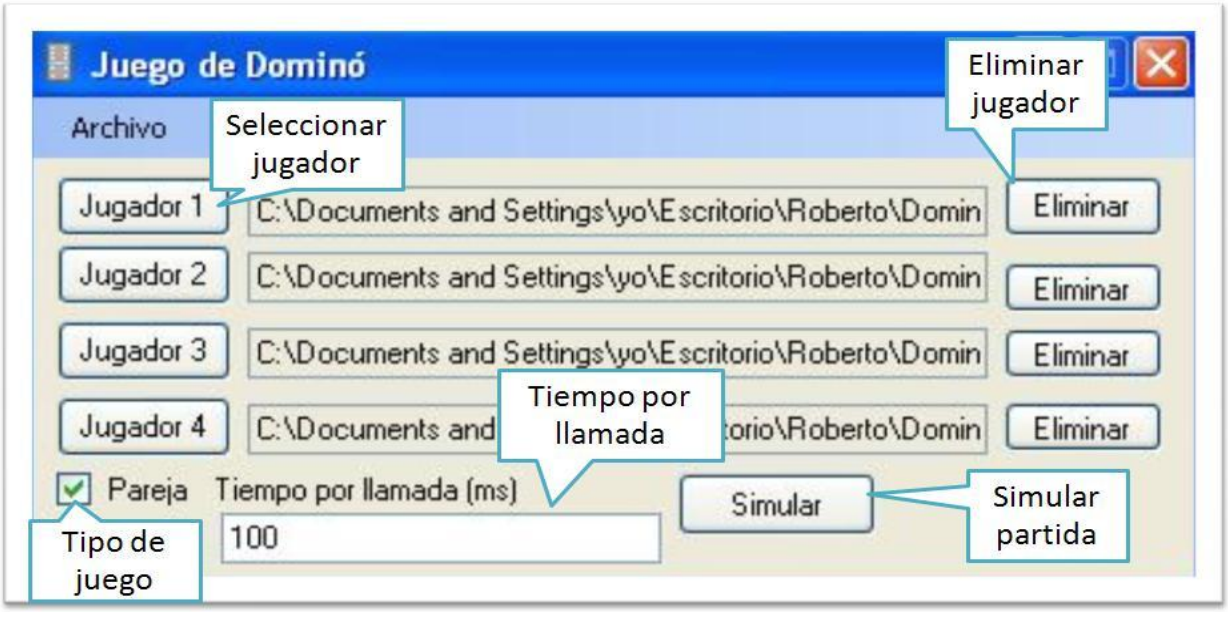

Tabla 2.6 Descripción del Caso de Uso: Jugar Partida

Caso de uso		
CU-3	Salvar Partida	
Propósito	El usuario desea salvar la partida realizada.	
Actores	Usuario	
Resumen	El caso de uso se inicia después de haberse simulado la partida, el usuario desea salvar la partida donde desee.	
Precondiciones	Que se haya realizado con éxito la simulación.	
Poscondiciones	Que se haya creado el archivo con el resultado de la misma.	
Acción del actor		**Respuesta del sistema**
1 El usuario después de haberse simulado la partida desea salvarla, para disponer de ella cuando quiera		1.1 El sistema muestra el cuadro de dialogo de salvar
2 El usuario entra los datos necesarios para salvar		2.1 El sistema guarda el archivo.
Pantalla		
		

Tabla 2.8 Descripción del Caso de Uso: Salvar Partida

Caso de uso		
CU- 2	Cargar Partida.	
Propósito	Cargar una partida.	
Actores	Usuario.	
Resumen	El caso de uso se inicia cuando el usuario desea cargar una partida.	
Precondiciones		
Acción del actor		**Respuesta del sistema**
1. El usuario solicita cargar una partida.		1.1 El sistema muestra un cuadro de dialogo de abrir fichero.
2. El usuario selecciona la partida a ver y la acepta.		2.1 El sistema carga el fichero y desencadena el caso de uso de Ver Partida
Pantalla		

Tabla 2.7 Descripción del Caso de Uso: Cargar Partida

Caso de uso		
CU-4	Ver Partida	
Propósito	El usuario desea ver la partida realizada.	
Actores	Usuario	
Resumen	El caso de uso se inicia después de haberse simulado la partida o haber solicitado ver partida.	
Precondiciones	Después de simular partida o cargar la partida este caso de uso es invocado.	
Poscondiciones		
Acción del actor		**Respuesta del sistema**
1 El usuario solicita ver la partida		1.1 El sistema muestra un formulario donde se muestra de forma gráfica el estado del juego. Brindando la posibilidad al usuario de ver la anterior y la siguiente acción y al concluir muestra el resultado de la partida.
Pantalla		

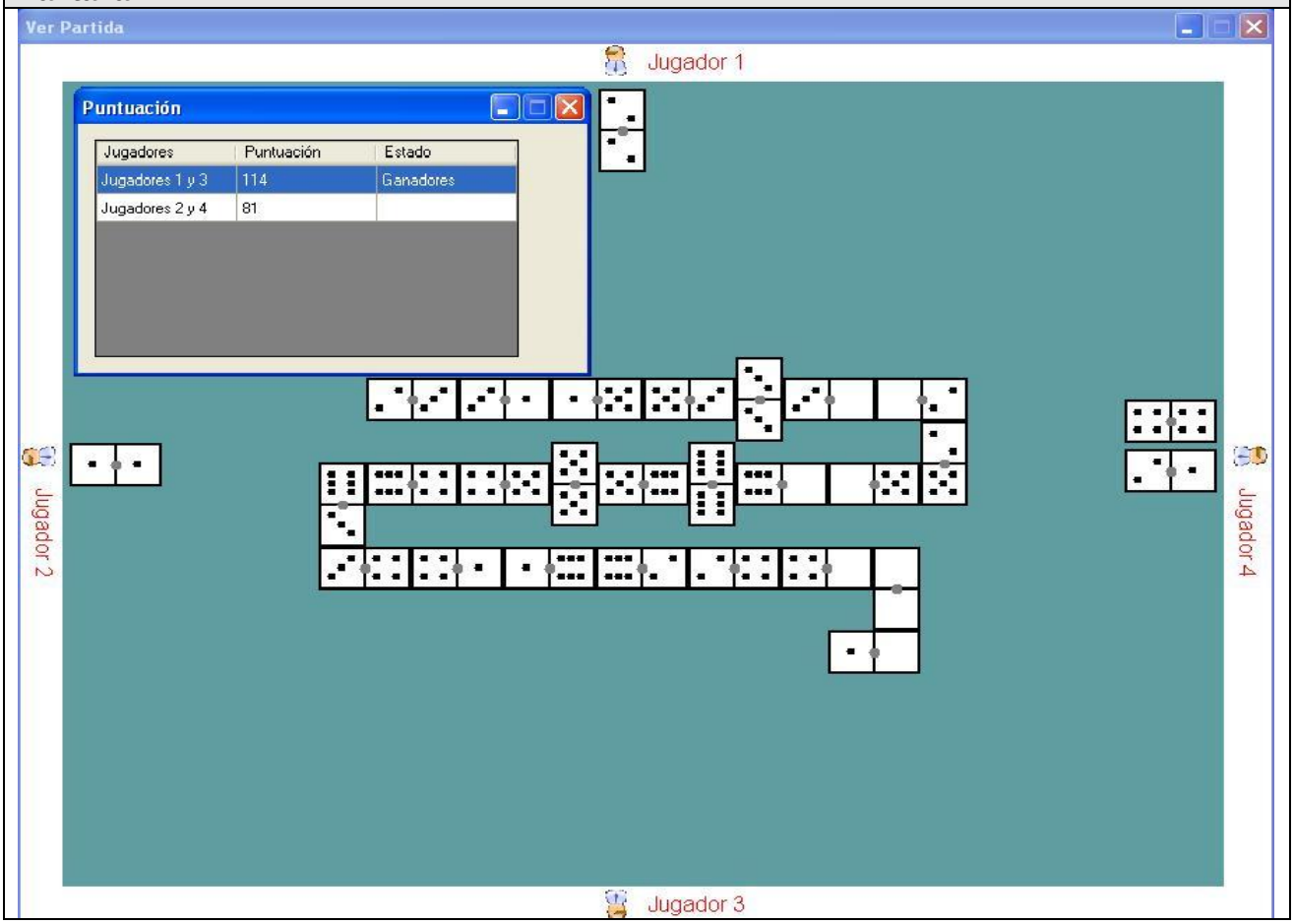

Tabla 2.9 Descripción del Caso de Uso: Ver Partida

Capítulo 3. Análisis y Diseño del Sistema

3.1 Introducción

En este capítulo se presentará los diagramas de clases del Análisis y los de Iteración del diseño. Además se mostrará el diagrama de clases del Diseño y las descripciones de las clases. También las descripciones de Aplicaciones gráficas para visualizar las partidas.

3.2 Diagramas de Clases del Análisis

Una clase del análisis representa una abstracción de un subsistema del diseño del sistema. Existen tres estereotipos (Interfaz, Control y Entidad) estandarizados en UML y se utilizan para ayudar a los desarrolladores a distinguir el ámbito de las diferentes clases.

3.2.1 Caso de uso: Jugar Partida.

(Ver Figura 3.1)

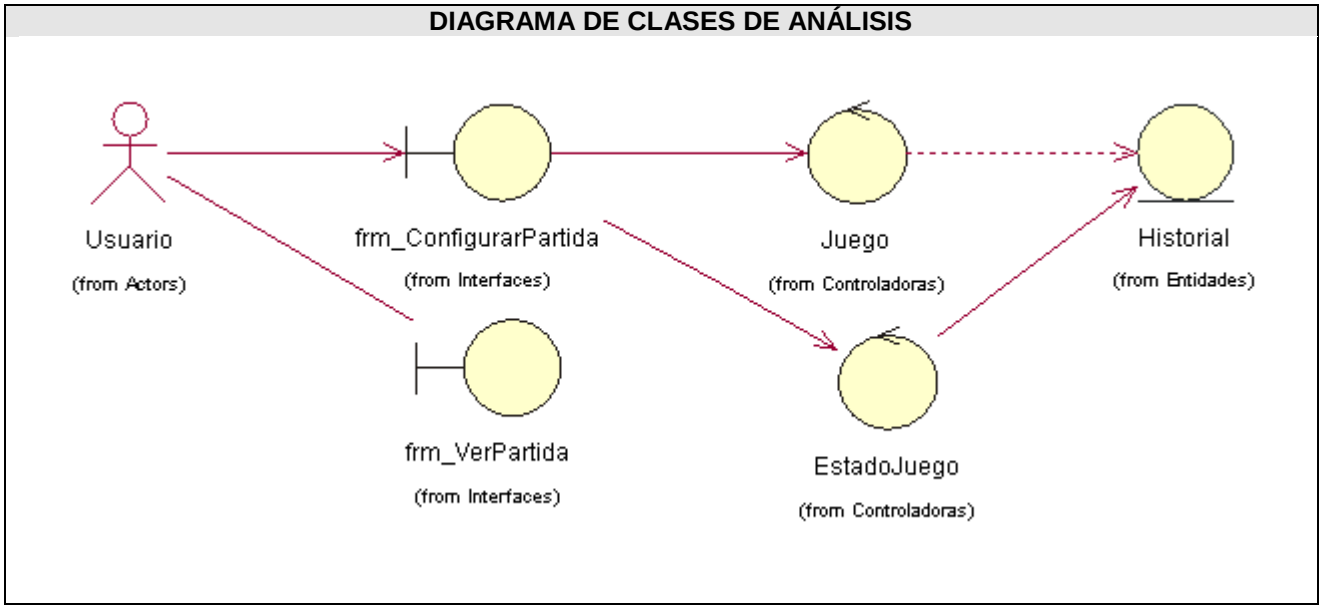

Figura 3.1 Caso de Uso: Jugar Partida

3.2.2 Caso de uso: Cargar Partida.

(Ver Figura 3.2)

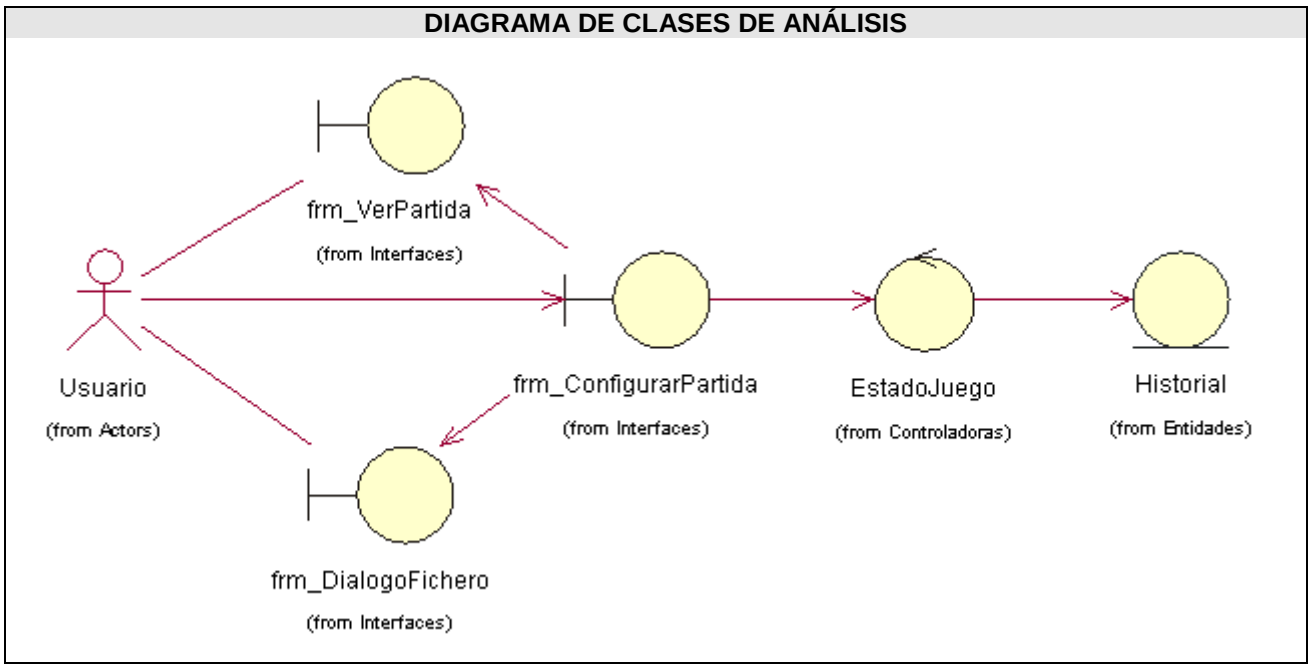

Figura 3.2 Caso de Uso: Cargar Partida

3.2.3 Caso de uso: Salvar Partida.

(Ver Figura 3.3)

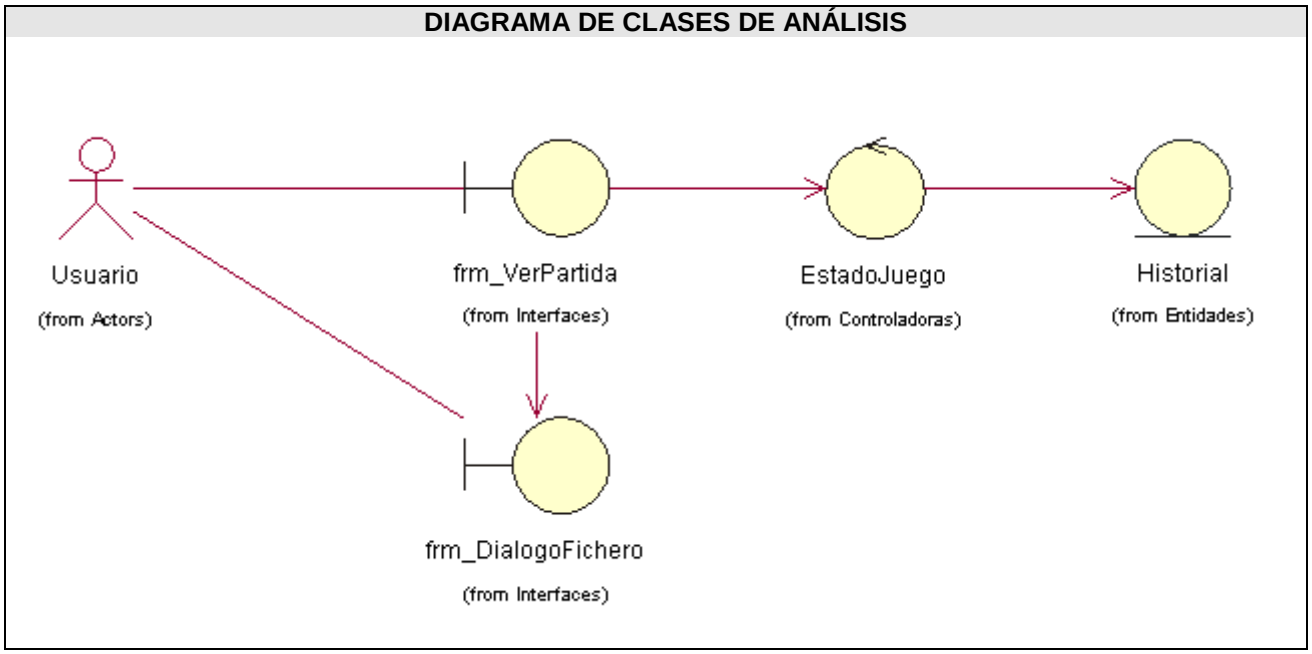

Figura 3.3 Caso de Uso: Salvar Partida

3.2.4 Caso de uso: Ver Partida.

(Ver Figura 3.4)

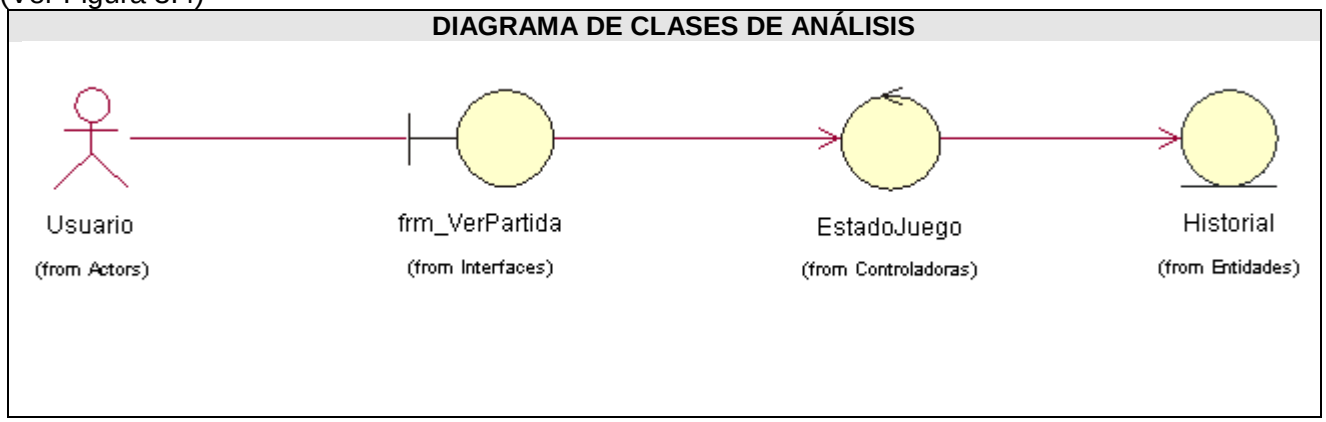

Figura 3.4 Caso de uso: Ver Partida

3.3 Diagramas de Interacción

Los diagramas de interacción no son más que una descripción del modo en el que cada operación detectada en los diagramas de secuencia lleva a cabo sus responsabilidades y modifica el estado del sistema. En UML los diagramas de interacción pueden representarse a través de los Diagramas de Colaboración y/o de los Diagramas de Secuencia.

Los diagramas de colaboración muestran las interacciones que ocurren entre los objetos que participan en una situación determinada. Esta es más o menos la misma información que la mostrada por los diagramas de secuencia, pero destacando la forma en que las operaciones se producen en el tiempo, mientras que los diagramas de colaboración fijan el interés en las relaciones entre los objetos y su topología.

En los diagramas de colaboración los mensajes enviados de un objeto a otro se representan mediante flechas, mostrando el nombre del mensaje, los parámetros y la secuencia del mensaje. Los diagramas de colaboración están indicados para mostrar una situación o flujo de programas específicos y son unos de los sobresalientes tipos de diagramas para demostrar o explicar rápidamente un proceso dentro de la lógica del programa.

3.3.1 Caso de Uso: Jugar Partida

(Ver Figura 3.1)

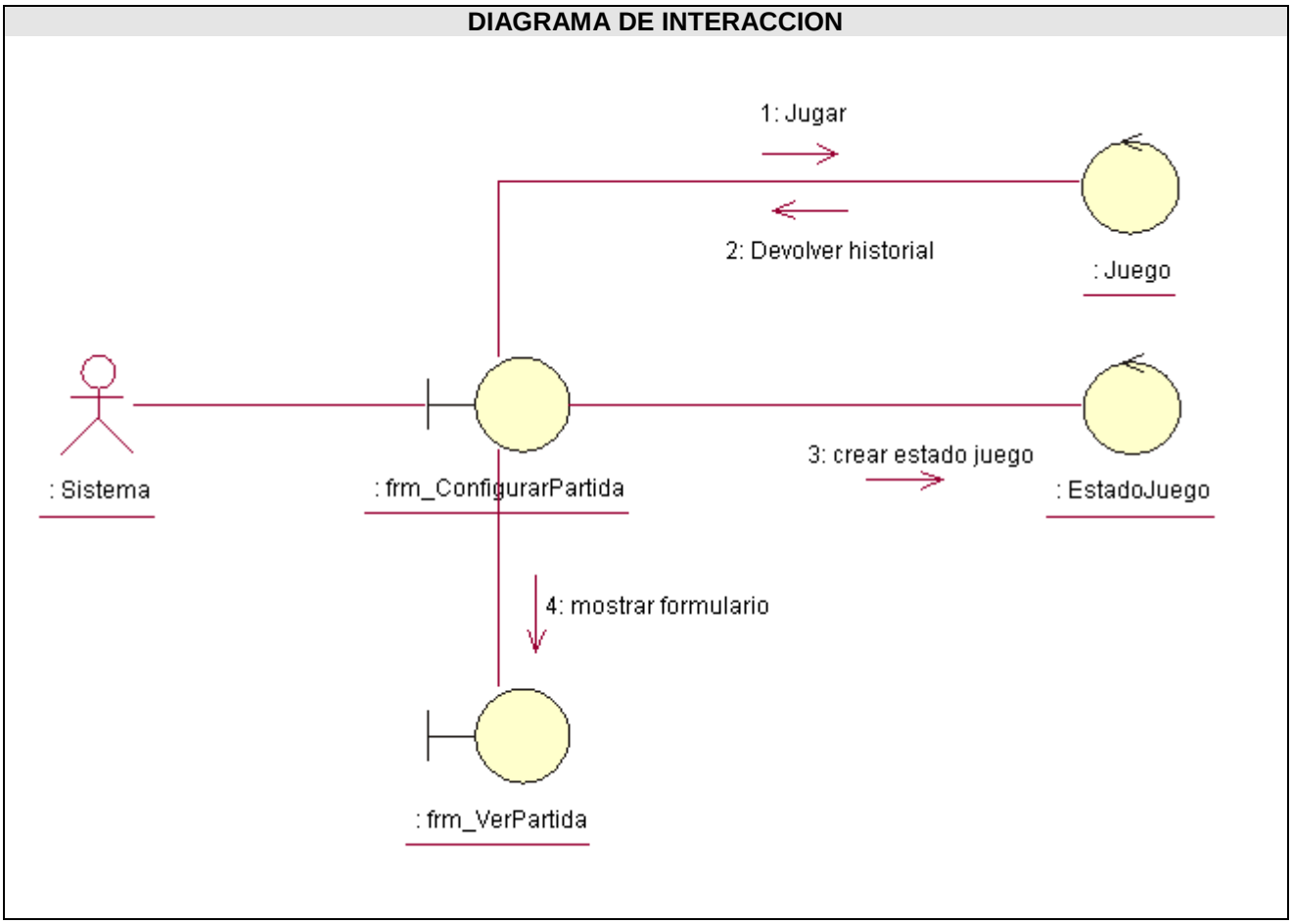

Figura 3.1 Caso de Uso: jugar Partida.

3.3.2 Caso de Uso: Cargar Partida.

(Ver Figura 3.2)

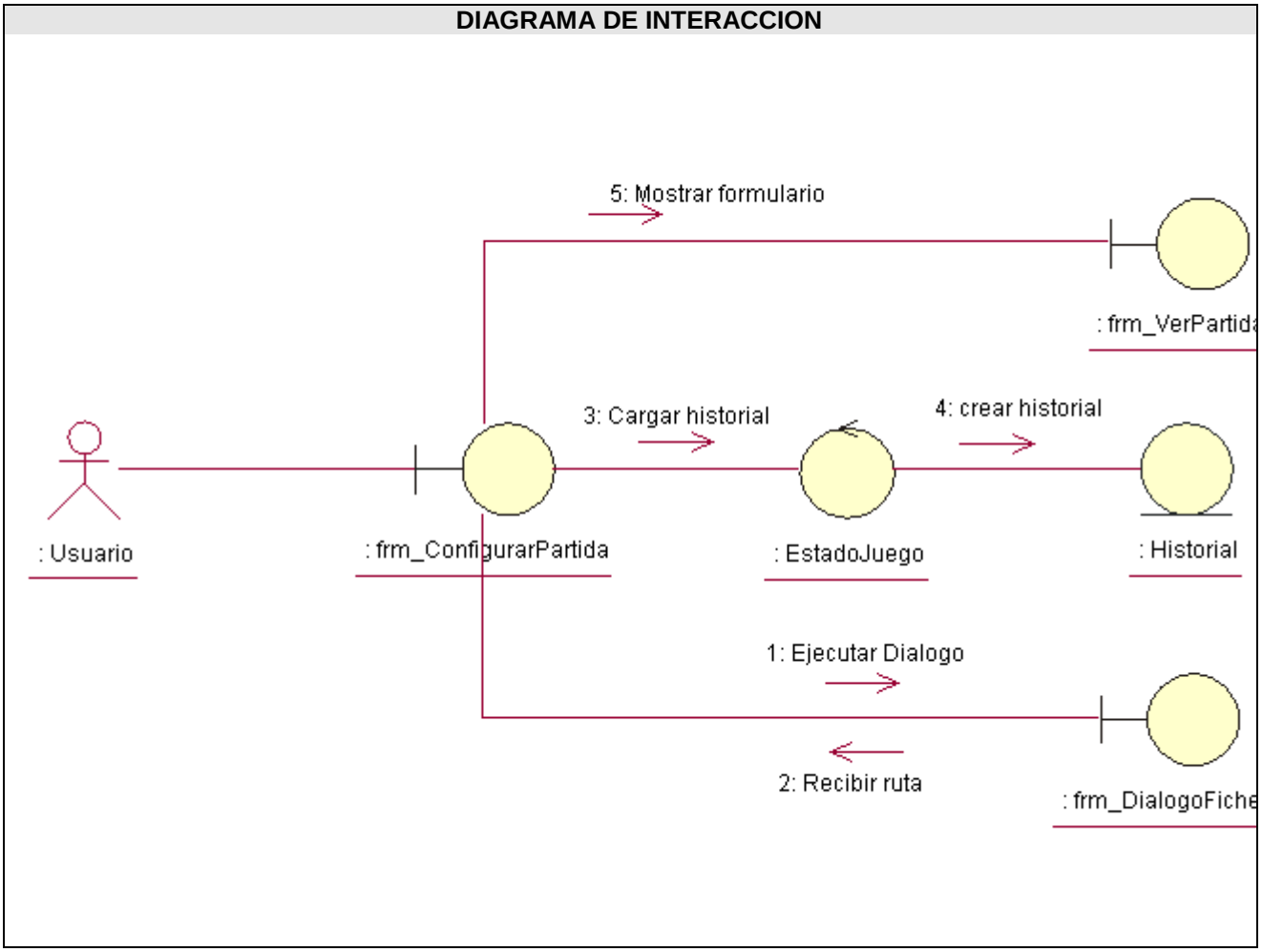

Figura 3.2 Caso de Uso: Cargar Partida

3.3.3 Caso de Uso: Salvar Partida.

(Ver Figura 3.3)

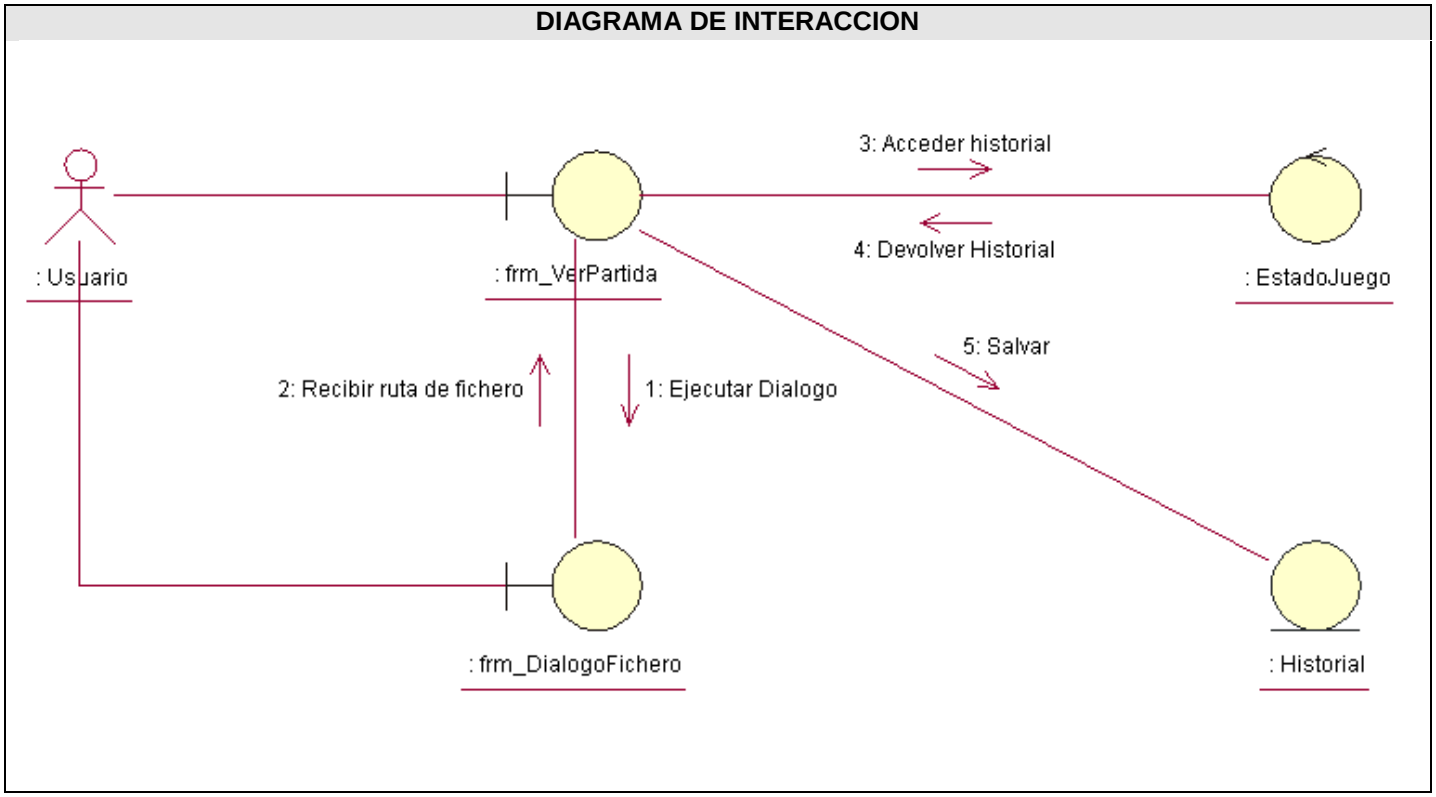

Figura 3.3 Caso de Uso: Salvar Partido.

3.3.4 Caso de Uso: Ver Partida.

(Ver Figura 3.4)

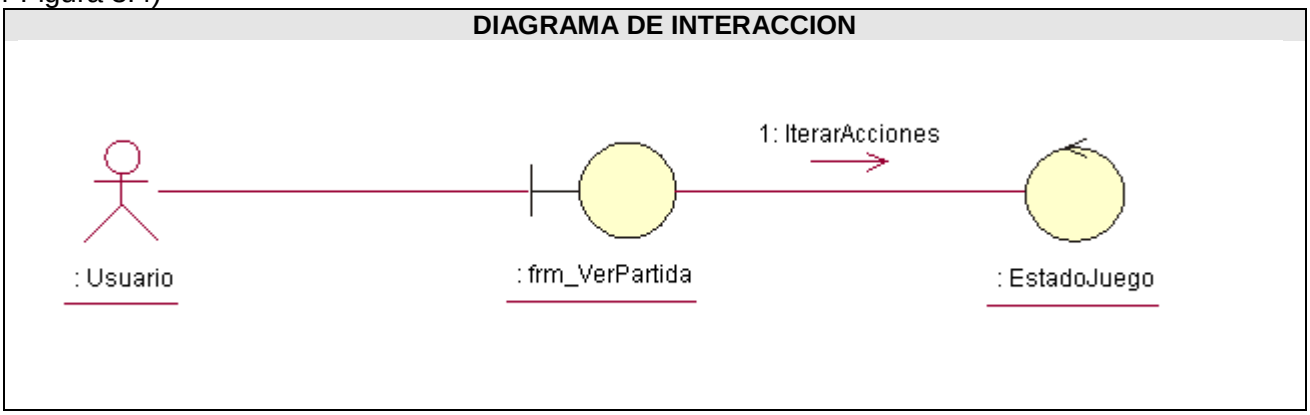

Figura 3.4 Caso de Uso: Ver Partida.

3.4 Diagramas de clases del diseño.

El Diagrama de clase del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. (Ver Figura 3.5 y 3.6)

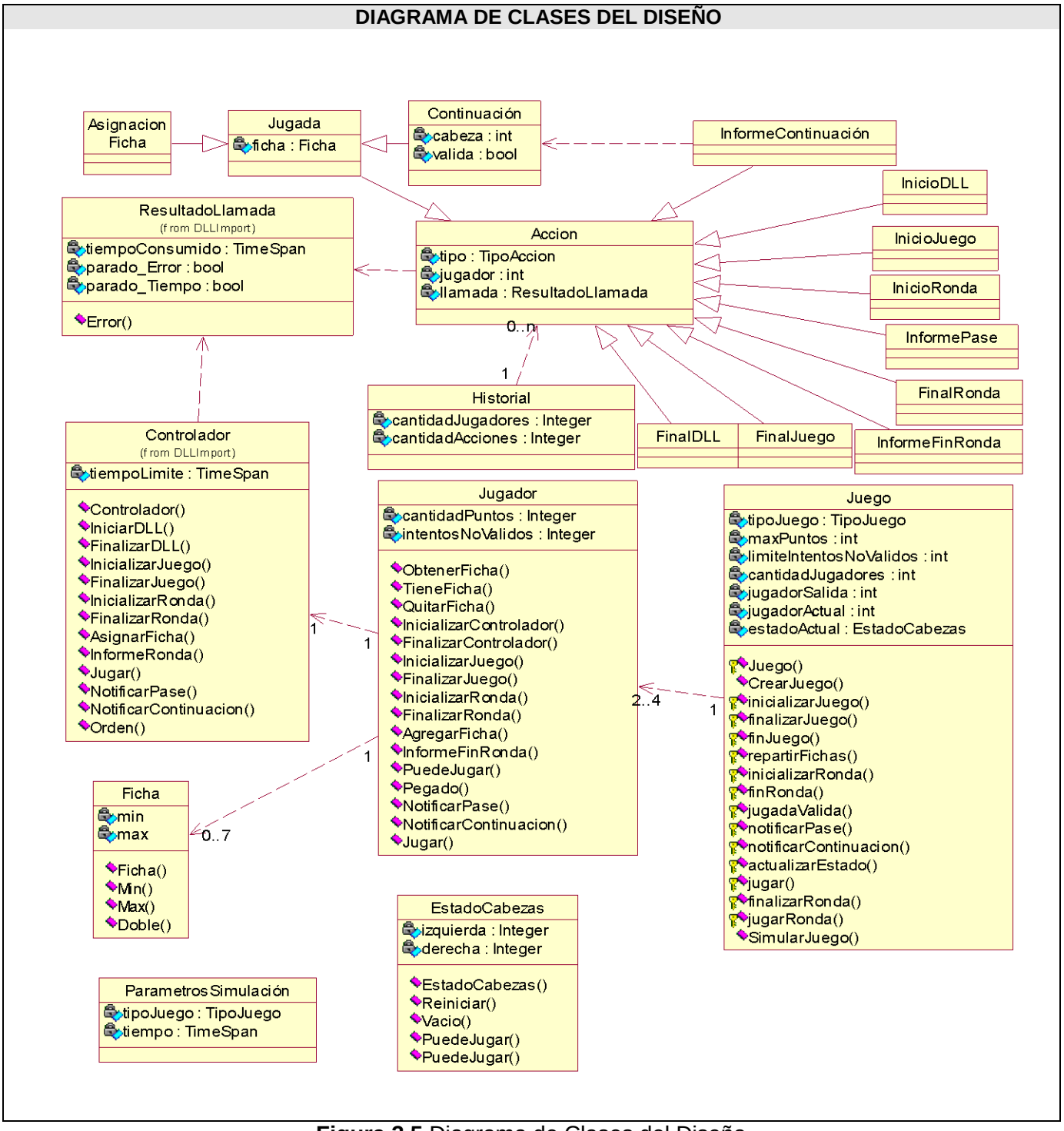

Figura 3.5 Diagrama de Clases del Diseño

3.4.1 Visor Gráfico

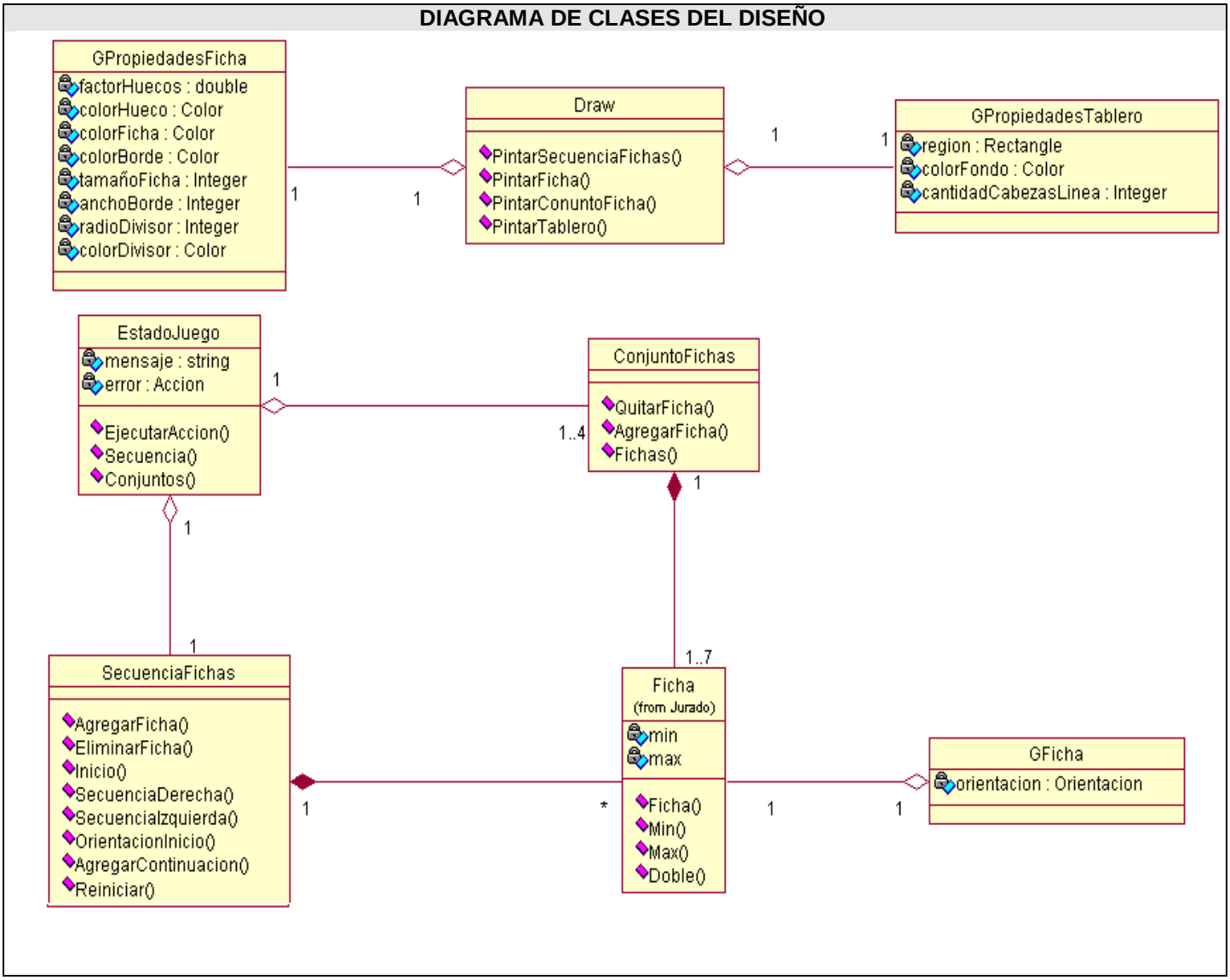

Figura 3.6 Diagrama de Clases del Diseño

3.5 Descripción de las Clases

En este epígrafe se realizará las descripciones textuales de las Clases del Diseño. Se mostrarán las más significativas (Ver Tabla 3.1, 3.2, 3.3, 3.4 y 3.5), las restantes aparecerán en los anexos. (Ver Anexos 1, 2, 3, 4, 5 y 6).

Nombre	DLLInterface
Tipo de clase	Controladora
Resumen	Esta clase es la encargada de llamar a los métodos de la DLL. Carga la DLL estáticamente.
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre	**Void IniciarDLL()**
Descripción	Llamar el métodos IniciarDLL en la DLL
Nombre	**Void FinalizarDLL()**
Descripción	Llamar el método FinalizarDLL en la DLL
Nombre	**void InicializarJuego**(int tipo)
Descripción	Llamar el método InicializarJuego en la DLL. Se le pasa el tipo de juego.
Nombre	**Void FinalizarJuego**()
Descripción	Llamar el método FinalizarJuego en la DLL
Nombre	**void InicializarRonda**()
Descripción	Llamar el método InicializarRonda en la DLL.
Nombre	**void FinalizarRonda**()
Descripción	Llamar el método FinalizarDLL en la DLL
Nombre	**Void AsignarFicha**(int min, int max)
Descripción	Llamar el método AsignarFicha en la DLL. La ficha es descrita por **min** y **max**
Nombre	**void InformeRonda**(int ganador, int puntos).
Descripción	Llamar el método **InformeRonda.** Se le pasa el Id ganador y la cantidad de puntos obtenidos.
Nombre	**Void Jugar**(int min, int max, bool cabeza)
Descripción	Llamar el método Jugar en la DLL. Esta devuelve el min y max de la ficha jugada y la cabeza por donde poner.
Nombre	**void NotificarPase**(int jugador).
Descripción	Llamar el método NotificarPase de la DLL. Este método notifica a la DLL acerca de un pase ocurrido. EL Id del jugador es especificado.
Nombre	**void NotificarContinuacion**(int jugador, int min, int max, bool cabeza).
Descripción	Llamar el método NotificarContinuacion. Se le pasa el jugador, el min y max de la ficha jugada y la cabeza por donde se jugo.
Nombre	Orden
Descripción	Obtener el Id del jugador. Coincide con el orden de jugadas.

Tabla 3.1 Descripción de la Clase: DLLInterface

Nombre	Controlador	
Tipo de clase	Controladora	
Resumen:	Esta clase es la encargada de llamar a los métodos de la DLL. Además de chequear el tiempo de llamada de estos. El tiempo de llamada esta ajustado por Tiempo_Limite.	
Atributo		**Tipo**
Tiempo_Limite		TimeSpan
Para cada responsabilidad:		
Nombre	**Controlador()**	
Descripción	Constructor del controlador	
Nombre	**ResultadoLlamada IniciarDLL()**	
Descripción	Llamar el métodos IniciarDLL en la interface de DLL	
Nombre	**ResultadoLLamada FinalizarDLL()**	
Descripción	Llamar el método FinalizarDLL en la interface de DLL	
Nombre	**ResultadoLLamada** InicializarJuego(int tipo, int id)	
Descripción	Llamar el método InicializarJuego en la interface de DLL. Se le pasa el tipo de juego y el índice asignado	
Nombre	**ResultadoLLamada FinalizarJuego**()	
Descripción	Llamar el método FinalizarJuego en la interface de DLL	
Nombre	**ResultadoLlamada InicializarRonda**()	
Descripción	Llamar el método InicializarRonda en la interface de DLL.	
Nombre	**ResultadoLlamada FinalizarRonda**()	
Descripción	Llamar el método FinalizarDLL en la interface de DLL	
Nombre	**ResultadoLlamada AsignarFicha**(int min, int max)	
Descripción	Llamar el método AsignarFicha en la interface de DLL. La ficha es descrita por **min** y **max**	
Nombre	**ResultadoLlamada InformeRonda**(int ganador, int puntos).	
Descripción	Llamar el método **InformeRonda** en la interface de DLL. Se le pasa el Id ganador y la cantidad de puntos obtenidos.	
Nombre	**ResultadoLlamada Jugar**(int min, int max, bool cabeza)	
Descripción	Llamar el método Jugar en la interface de DLL. Esta devuelve el min y max de la ficha jugada y la cabeza por donde poner.	
Nombre	**ResultadoLlamada NotificarPase**(int jugador).	
Descripción	Llamar el método NotificarPase en la interface de DLL. Este método notifica a la DLL acerca de un pase ocurrido. EL Id del jugador es especificado.	
Nombre	**ResultadoLamada NotificarContinuacion**(int jugador, int min, int max, bool cabeza).	
Descripción	Llamar el método NotificarContinuacion en la interface de DLL. Se le pasa el jugador, el min y max de la ficha jugada y la cabeza por donde se jugo.	
Nombre	Orden	
Descripción	Obtener el Id del jugador. Coincide con el orden de jugadas.	

Tabla 3.2 Descripción de la Clase: Controlador

Nombre	Ficha	
Tipo de clase	Entidad	
Resumen	Esta clase tiene la responsabilidad de guardar los datos de una ficha.	
Atributo		**Tipo**
min		integer
max		integer
Para cada responsabilidad:		
Nombre	**Ficha(int min, int max)**	
Descripción	Constructor de la ficha. Especificando el maximo y el minimo.	
Nombre	**Int Min()**	
Descripción	El valor de la cabeza minima.	
Nombre	**Int Max()**	
Descripción	El valor de la cabeza maxima	
Nombre	**Bool Doble()**	
Descripción	Restorna si la ficha es un doble o no.	

Tabla 3.3 Descripción de la Clase: Ficha

Nombre	Jugador	
Tipo de clase	Controladora	
Resumen	Esta clase abstrae un jugador y simula el comportamiento y estado de un jugador en el juego.	
Atributo		**Tipo**
cantidadPuntos		integer
intentosNoValidos		integer
Para cada responsabilidad:		
Nombre	**Ficha ObtenerFicha(int n)**	
Descripción	Obtener la ficha enésima que posee el jugador en el estado actual.	
Nombre	**Bool TieneFicha(Ficha ficha)**	
Descripción	Saber si el jugador posee la ficha especificada en el estado actual.	
Nombre	**void QuitarFicha(Ficha ficha)**	
Descripción	Eliminar la ficha especificada del jugador.	
Nombre	**InicioDLL IniciarControlador()**	
Descripción	Inicializar el controlador del jugador.	
Nombre	**FinalDLL FinalizarControlador()**	
Descripción	Finalizar el controlador del jugador.	
Nombre	**InicioJuego InicializarJuego()**	
Descripción	Inicializar el Juego para el jugador.	
Nombre	**FinalJuego FinalizarJuego()**	
Descripción	Finalizar el Juego para el jugador.	
Nombre	**InicioRonda InicializarRonda()**	
Descripción	Inicializar la ronda para el jugador.	
Nombre	**FinalRonda FinalizarRonda()**	
Descripción	Finalizar la ronda para el jugador.	
Nombre	**AsignaciónFicha AsignarFicha(Ficha ficha)**	
Descripción	Asignar la ficha al jugador.	
Nombre	**InformeFinRonda InformarFinRonda(int ganador, int puntos)**	
Descripción	Notificar al jugador acerca del fin de ronda y sus resultados.	
Nombre	**Bool PuedeJugar(EstadoCabezas estado)**	

Descripción	Retorna si el jugador en el estado actual puede jugar con el estado de las cabezas especificado.
Nombre	**Bool Pegado()**
Descripción	Notifica si el jugador esta pegado o no.
Nombre	**Bool Pegado()**
Descripción	Notifica si el jugador esta pegado o no.
Nombre	**InformePase NotificarPase(int jugador)**
Descripción	Notificar el pase al jugador.
Nombre	**InformeContinuación NotificarContinuacion(Continuación continuacion)**
Descripción	Notifica la continuación de una jugada.
Nombre	**Continuación Jugar()**
Descripción	Pone al jugador a jugar y devuelve la jugada hecha.

Tabla 3.4 Descripción de la Clase: Jugador

Nombre	Juego	
Tipo de clase	Controladora	
Resumen	Esta clase abstrae un jugador y simula el comportamiento y estado de un jugador en el juego.	
Atributo		**Tipo**
tipoJuego		TipoJuego
maxPuntos		integer
limiteIntentosNoValidos		int
cantidadJugadores		int
jugadorSalida		int
jugadorActual		int
estadoActual		EstadoCabezas
Para cada responsabilidad:		
Nombre	**Juego()**	
Descripción	Constructor del objeto.	
Nombre	**Juego CrearJuego(ParametrosSimulación parametros, string carpeta)**	
Descripción	Crea un Juego pasando los parámetros de simulación y la carpeta donde debe buscar las DLLs.	
Nombre	**bool inicializarJuego(ArrayList list) : bool**	
Descripción	Inicializa el juego y devuelve el arreglo de Acciones ocurridas. Devuelve si hubo error.	
Nombre	**bool finalizarJuego**	
Descripción	Finaliza el juego y devuelve el arreglo de Acciones ocurridas. Devuelve si hubo error.	
Nombre	**bool finJuego()**	
Descripción	Determina si el juego ha llegado a su final o no	
Nombre	**bool repartirFichas(ArrayList list)**	
Descripción	Reparte las fichas a los jugadores de manera aleatoria y devuelve la lista de acciones ocurridas. Devuelve también si hubo error.	
Nombre	**bool inicializarRonda(ArrayList list)**	
Descripción	Inicializa una ronda y devuelve la lista de acciones ocurridas. Devuelve si hubo error.	
Nombre	**Bool finalRonda()**	
Descripción	Determina si la ronda ha llegado al final.	
Nombre	**bool jugadaValida(Ficha ficha, bool cabeza)**	
Descripción	Determina si la ficha especificada se puede poner por la cabeza especificada con el estado actual del tablero.	
Nombre	**bool notificarPase (ArrayList list)**	
Descripción	Notifica un pase a los jugadores y devuelve la lista de acciones realizadas. Devuelve si hubo error.	

Nombre	**Bool notificarContinuacion (Continuacion continuacion , ArrayList list)**
Descripción	Notifica la continuación a los jugadores. Devuelve la lista de acciones ocurridas y si hubo error.
Nombre	**Void actualizarEstado(continuación : Continuación)**
Descripción	Actualiza el estado actual del tablero teniendo en cuenta la continuación especificada.
Nombre	**bool jugar(ArrayList list)**
Descripción	Hace jugar el jugador en turno y devuelve las acciones ocurridas. Devuelve si hubo error.
Nombre	**bool finalizarRonda (ArrayList list)**
Descripción	Finaliza una ronda y devuelve las acciones realizadas y si hubo error.
Nombre	**bool : jugarRonda (ArrayList list)**
Descripción	Juega una ronda completa y retorna las acciones ocurridas. Devuelve si hubo error.
Nombre	**Historial SimularJuego()**
Descripción	Juega una partida completa y devuelve el historial completo de juego.

Tabla 3.5 Descripción de la Clase: Jugar

3.5.1 Descripciones de las aplicaciones gráficas.

Se mostrará en este epígrafe las descripciones de las aplicaciones gráficas para visualizar las partidas. Solo aparecerán las tablas de las clases más significativas (Ver Tablas 3.6, 3.7, 3.8 y 3.9), las restantes se encontrarán en los anexos. (Ver Anexos 7, 8 y 9)

Nombre	GPropiedadesFicha
Tipo de clase	Entidad
Resumen	Esta clase guarda las propiedades gráficas de las fichas.
Atributo	**Tipo**
factorHuecos	double
colorHueco	Color
colorFicha	Color
colorBorde	Color
anchoBorde	int
radioDivisor	int
colorDivisor	Int

Tabla 3.6 Descripción de la Aplicación Gráfica: GPropiedadesFicha

Nombre	GPropiedadesTablero
Tipo de clase	Entidad
Resumen	Esta clase guarda las propiedades gráficas del tablero.
Atributo	**Tipo**
region	double
colorFondo	Color
cantidadCabezasLinea	Color

Tabla 3.7 Descripción de la Aplicación Gráfica: GPropiedadesTablero

Nombre	GFicha
Tipo de clase	Entidad
Resumen	Esta clase guarda la información necesaria para pintar una ficha
Atributo	**Tipo**
Orientación	Orientación : (minimoIzquierda, minimoDerecha, minimoArriba, minimoAbajo)

Tabla 3.8 Descripción de la Aplicación Gráfica: GFicha

Nombre	ConjuntoFichas
Tipo de clase	Controladora
Resumen	Esta clase contiene la información de las fichas que tiene un jugador.

Atributo	**Tipo**
Para cada responsabilidad:	
Nombre	**void QuitarFicha(Ficha ficha)**
Descripción	Quitar la ficha
Nombre	**void AgregarFicha(Ficha ficha)**
Descripción	Agregar una ficha
Nombre	**Ficha[ ] Fichas()**
Descripción	Obtener el arreglo de fichas actuales

Tabla 3.9 Descripción de la Aplicación Gráfica: ConjuntoFichas

Capítulo 4. Implementación

4.1 Introducción

En el siguiente Capítulo se modelará el diagrama de despliegue y el diagrama de componente.

4.2 Diagrama de despliegue.

En el diagrama de despliegue es un modelo de objeto que describe la distribución física del sistema, representa la arquitectura de tiempo de ejecución de los procesadores y dispositivos, permite apreciar de forma visual cómo se encuentra relacionados físicamente los componentes de la aplicación. En este caso el sistema solo depende de un dispositivo (PC) para realizar su ejecución. (Ver Figura 4.1)

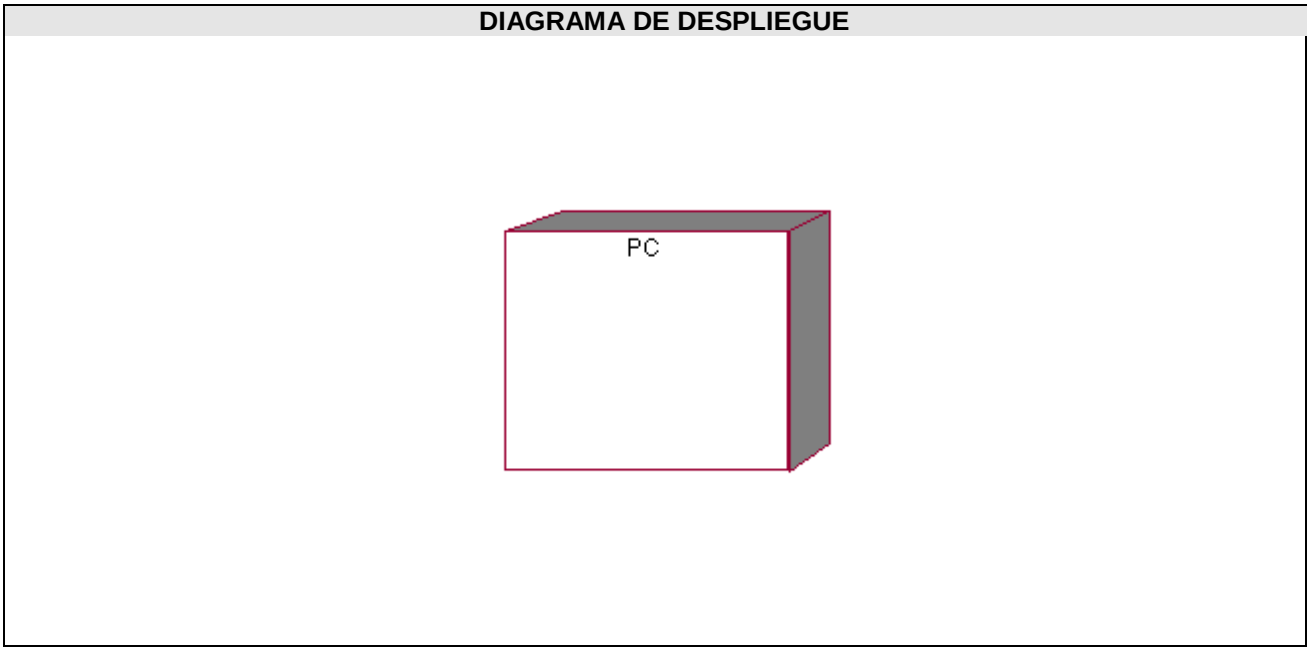

Figura 4.1 Diagrama de Despliegue

4.3 Diagrama de componentes.

El diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes.

Un paquete en un diagrama de componentes representa una división física del sistema. Los paquetes se organizan en una jerarquía de capas donde cada capa tiene una interfaz bien definida.

A continuación se presenta la vista general del diagrama de componentes, el cual se ha dividido en cuatro paquetes (Ver Figura 4.2), el diagrama de cada uno de los paquetes se muestra también a continuación (Ver Figura 4.3, 4.4 y 4.5).

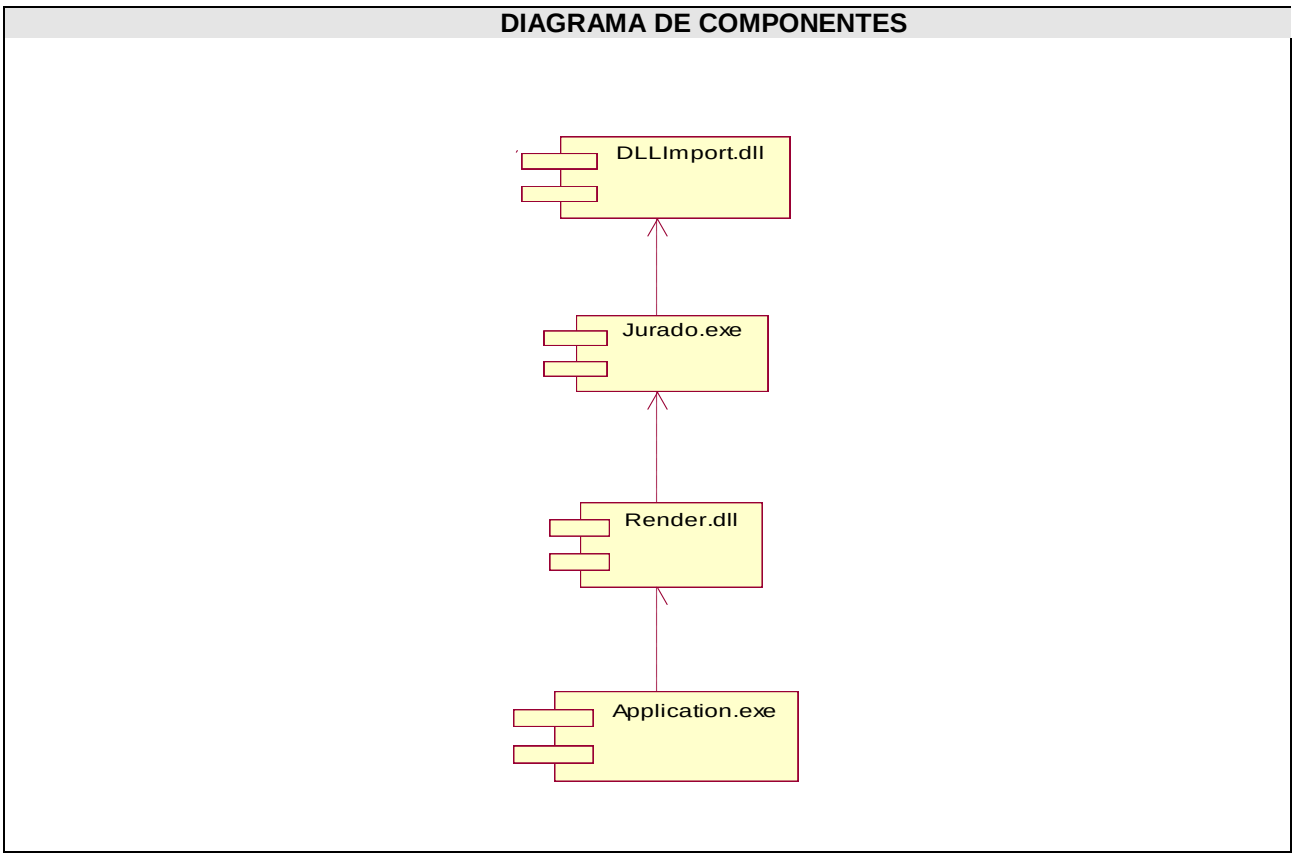

Figura 4.2 Diagrama de Componentes

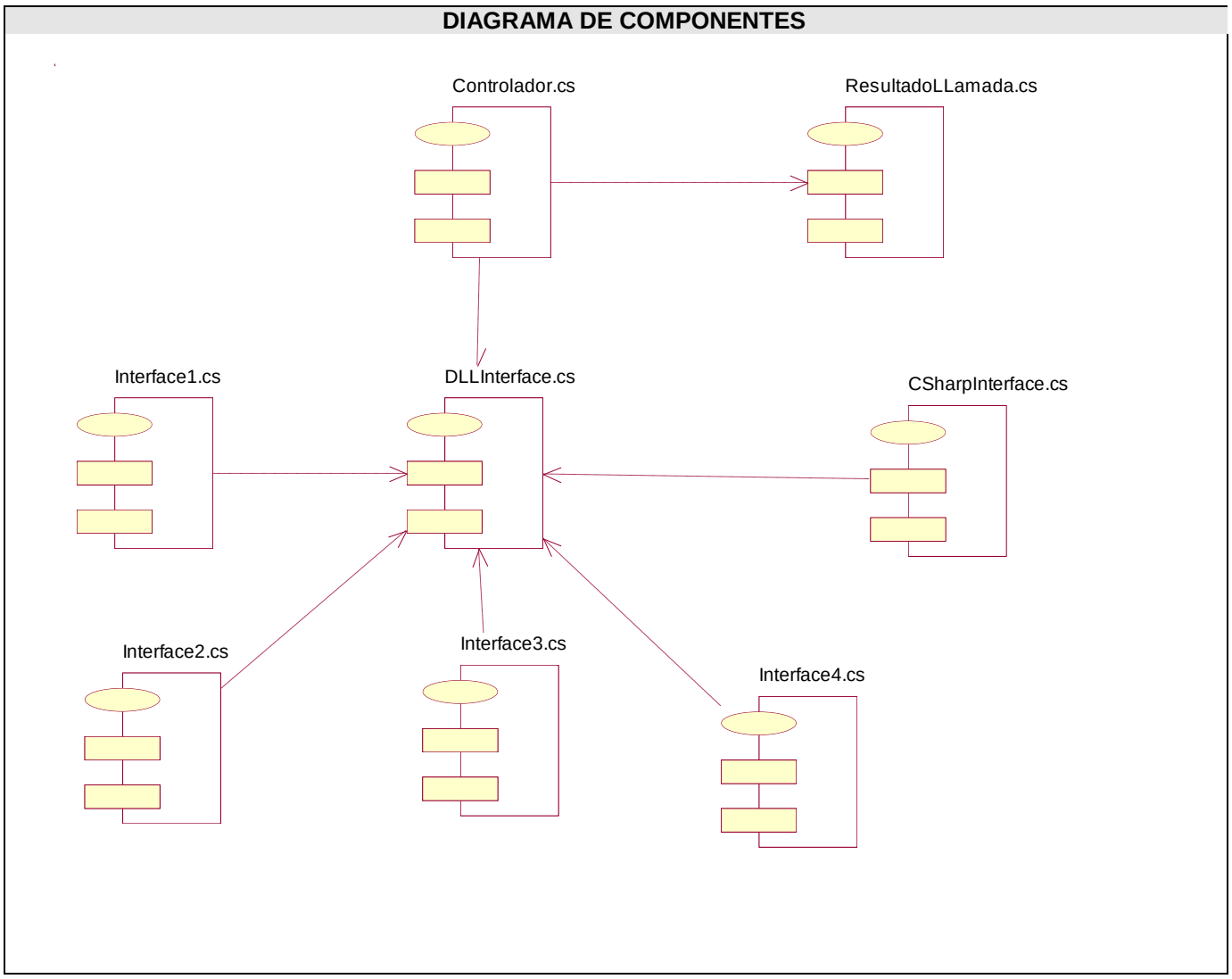

Figura 4.3 Diagrama de componentes: DLLImport.dll

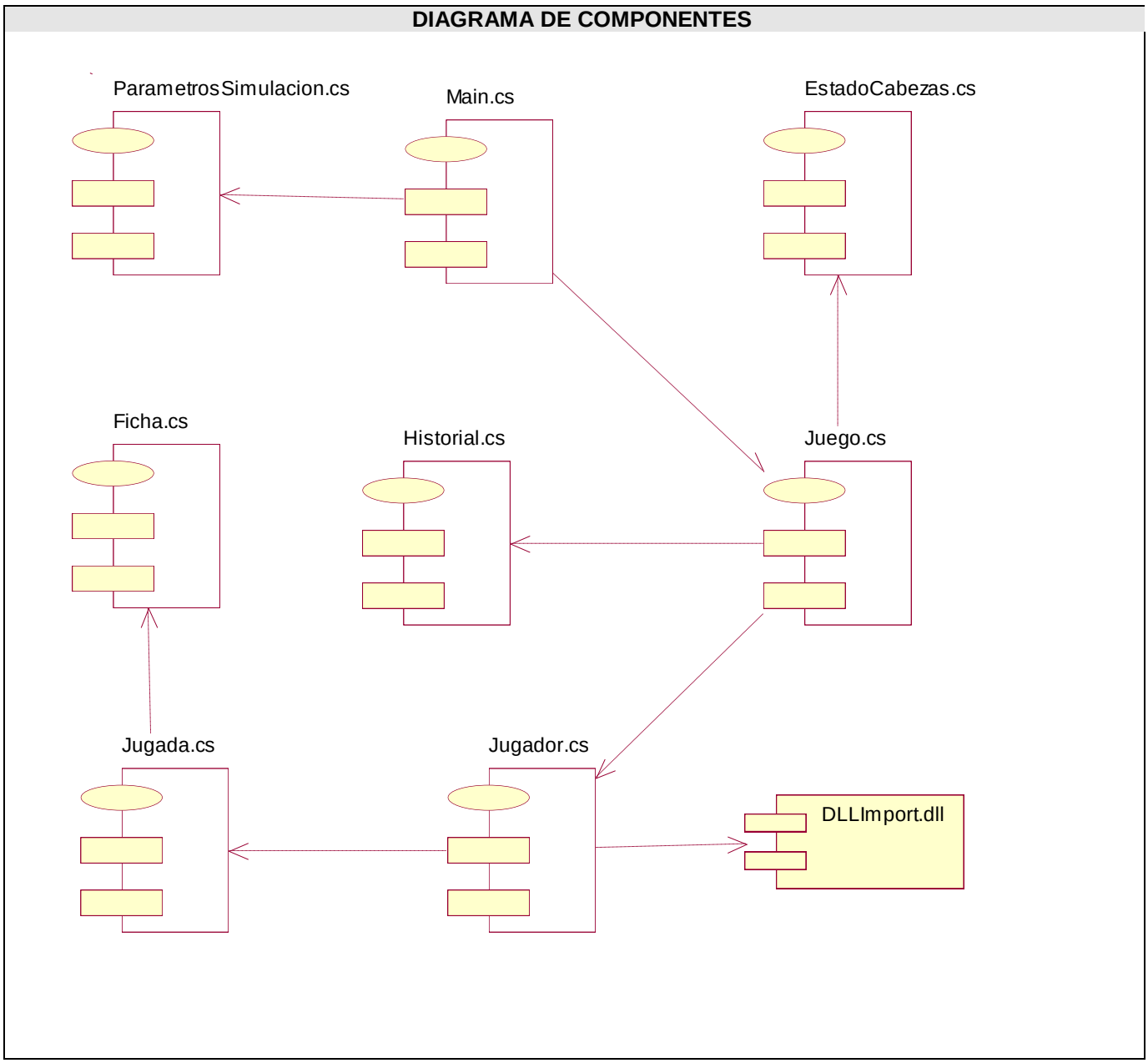

Figura 4.4 Diagrama de componentes: Jurado.exe

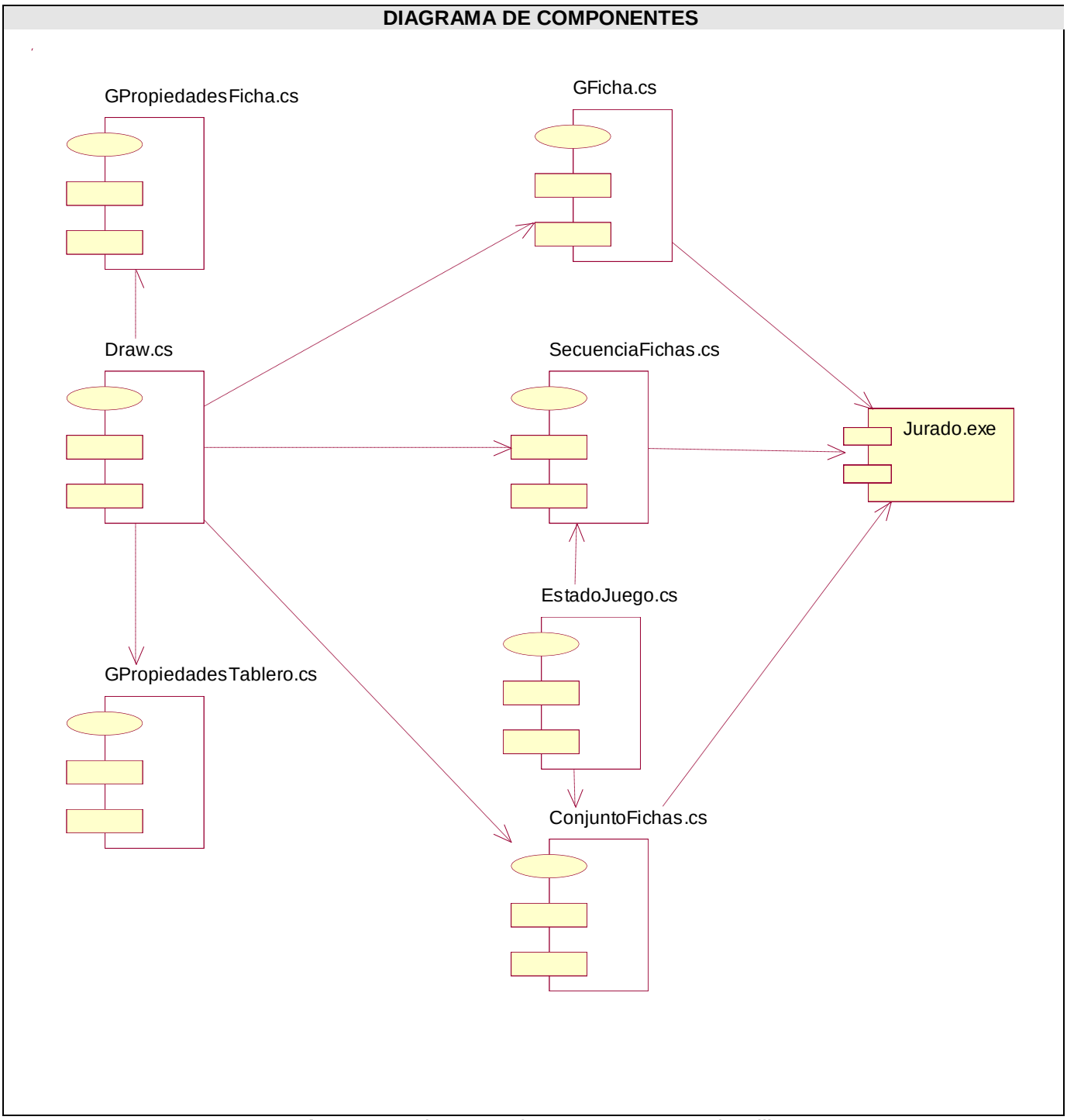

Figura 4.5 Diagrama de compontes: Render.dll

Conclusiones

Con la culminación de este trabajo, se pudo comprobar la gran variedad de juegos virtuales existentes en el mundo y su aceptación por parte de cualquier usuario. Se comprobó que la mayoría de estos juegos influyen en las mentes de las personas y que en ocasiones a cambiado hasta el concepto del aprendizaje.

Al término del presente trabajo de diploma, se logró:

- La realización de un juego de dominó para jugadores virtuales.

- Que los estudiantes, profesores y todo aquel interesado de la Universidad de las Ciencias Informáticas disponga de una nueva opción más amena y dinámica para la práctica de las técnicas de la programación.

- Ampliar el universo de juegos virtuales existentes en la Universidad donde los estudiantes vinculen sus conocimientos con las actividades lúdicas.

Recomendaciones

- En futuros trabajos se puede hacer un jurado mucho más enriquecido con tipos de juegos distintos que tengan el dominó como base. Además de esto la arquitectura esta pensada para que el jurado sea independiente de la aplicación que visualiza la partida, con el propósito de que se puedan hacer nuevas aplicaciones para otros propósitos.

- Con el propósito de poder reutilizar las funcionalidades del jurado se proveen junto con la aplicación una serie de componentes para facilitar su construcción:
    - **DLLImport.dll:** Conjunto de clases para cargar y manipular los métodos de los jugadores.
    - **Jurado.Exe:** Conjunto de clases que permite hacer todo el trabajo de simular una partida completa entre los jugadores virtuales.
    - **Render.dll:** Conjunto de clases que permite dar soporte gráfico al juego.

  Usando estos componentes se podrían llegar a hacer nuevas aplicaciones con otros propósitos para jugadores virtuales de dominó.

- Que en futuras aplicaciones se pueda desarrollar este tipo de juego, donde puedan participar jugadores virtuales de forma online.

- Darle seguimiento al estudio a este tipo de aplicaciones para jugadores virtuales debido a que en la universidad se ha incursionado muy poco en el tema tratado.

Bibliografía

Master Magazine. Visual Studio .NET, 2007, [Disponible http://www.mastermagazine.info/articulo/2086.php]

Wendy Chen. Los Juegos Virtuales hoy son cosas de mayores, 2008. [Disponible http://www.bjinforma.com/2004-02/2004.02-china-1.htm]

Castro-Perea. Los juegos virtuales se confundirán con la realidad, 2008. [Disponible http://www.tendencias21.net/En-dos-anos,-los-juegos-virtuales-se-confundiran-con-la-realidad_a1457.html]

Anónimo. El lenguaje de programación C#, 2007, [Disponible http://www.agapea.com/El-lenguaje-de-programacion-C--n11766i.htm]

Anónimo. Metodología (ingeniería de software), 2007. [Disponible http://es.wikipedia.org/wiki/Metodolog%ADa]

Learning. C: Tutorial de Lenguaje C, 2008. [Disponible http://members.cox.net/midian/articles/ansic1.htm]

Steve Holmes, C Programming, 2008. [Disponible http://www.its.strath.ac.uk/courses/c/]

Wikibooks: "C++ Programming", 2007 [Disponible http://en.wikibooks.org/wiki/C%2B%2B_Programming]

Bruce Eckel "Thinking in C++" 2ed. 2008 [Disponible http://mindview.net/Books/TICPP/ThinkingInCPP2e.html]

Nicholas A. Solter, Scott J. Kleper "Professional C++ (Programmer to Programmer)". Wrox Books. 2005. ISBN: 0764574841

Wikipedia: "Comparison of programming languages", 2008. [Disponible http://en.wikipedia.org/wiki/Comparisonofprogramming_languages]

Ian Joyner: "A Critique of C++" (3rd Ed.), 2007. [Disponible http://burks.bton.ac.uk/burks/pcinfo/progdocs/cppcrit/]

Bjarne Stroustrup "Learning C++ as a New Language", 2008. [Disponible http://www.research.att.com/~bs/new_learning.pdf]

GLib Reference Manual, 2007. [Disponible http://developer.gnome.org/doc/API/2.0/glib/]

Xerces C++ Parser (XML Library), 2008. [Disponible http://xml.apache.org/xerces-c/]

Boost C++ Libraries, 2007. [Disponible http://www.boost.org/]

Anónimo. Microsoft Visual Studio .NET 2003 Beta Final Una revisión del entorno actual: ¿vale la pena actualizarse?, 2007, [Disponible http://www.idg.es/pcworld/index.asp?link=estructura/i_articulo_centroArticulo.asp&IdArticulo=146672]

Anónimo. Origen y necesidad de un nuevo lenguaje, 2007, [Disponible http://recursos.dotnetclubs.com/sevilla/2005-06/Aportaciones/Libro%20sobre%20C%23.pdf]

Santiago Lamelo Fagilde, Análisis de la V.O., 2008. [Disponible http://ac2.turbinegames.com/]

Robert Cailliau, WorldWideWeb, 2008. [Disponible http://es.wikipedia.org/wiki/World_Wide_Web]

Referencias Bibliográficas

"Biblioteca de vínculos dinámicos." Microsoft® Student 2008 [DVD]. Microsoft Corporation, 2007.

"Lenguaje de consulta estructurado." Microsoft® Student 2008 [DVD]. Microsoft Corporation, 2007.

"Internet." Microsoft® Student 2008 [DVD]. Microsoft Corporation, 2007

"Inteligencia artificial." Microsoft® Student 2008 [DVD]. Microsoft Corporation, 2007.

"C#." Microsoft® Student 2008 [DVD]. Microsoft Corporation, 2007.

"C++." Microsoft® Student 2008 [DVD]. Microsoft Corporation, 2007.

"Lenguaje de programación." Microsoft® Student 2008 [DVD]. Microsoft Corporation, 2007

Roger S. Pressman. Un enfoque práctico. Parte 1. La Habana, Editorial Félix Varela, 2005, Pág. 601.

Roger S. Pressman. Un enfoque práctico. Parte 2. La Habana, Editorial Félix Varela, 2005, Pág. 600.

Craig Larman. UML y Patrones: Introducción al análisis y diseño orientado a objetos. Parte 1. La Habana, Editorial Félix Varela, 2004. Pág. 290.

Craig Larman. UML y Patrones: Introducción al análisis y diseño orientado a objetos. Parte 2. La Habana, Editorial Félix Varela, 2004. Pág. 300.

Anexos

Anexo 1

Descripción de la Clase: ResultadoLlamada

Nombre	**ResultadoLlamada**
Tipo de clase	**Entidad**
Resumen	Esta clase es la encargada de recoger los datos de resultados de la llamada a la DLL.

Atributo	**Tipo**
tiempoConsumido	TimeSpan
parado_Error	bool
parado_Tiempo	bool
Para cada responsabilidad:	
Nombre:	**Bool Error()**
Descripción:	Devuelve si hubo algun errror, de tiempo o de ejecución.

Anexo 2

Descripción de la Clase: CSharpInterface

Nombre	CSharpInterface
Tipo de clase	Controladora
Resumen:	Cuando se trabaja con una DLL de C# se puede importar esta clase por reflextion. Al implementar la DLL es necesario heredar de esta clase.

Atributo	**Tipo**
orden	int

Anexo 3

Descripción de la Clase: ParametrosSimulación

Nombre	ParametrosSimulación
Tipo de clase	Entidad
Resumen	Esta clase almacena los parámetros de la simulación de la partida.
Atributo	**Tipo**
tipoJuego	TipoJuego
tiempo	TimeSpan

Anexo 4

Descripción de la Clase: EstadoCabezas

Nombre	EstadoCabezas
Tipo de clase	Entidad
Resumen	Esta clase tiene la responsabilidad de guardar el estado actual del tablero por ronda.
Atributo	**Tipo**
izquierda	integer
derecha	integer
Para cada responsabilidad:	
Nombre	**Void Reiniciar()**
Descripción	Reiniciar el estado del tablero. Para iniciar una ronda.
Nombre	**Bool Vacio().**
Descripción	Para saber si el tablero esta vacío. Para saber si hay que sacar.
Nombre	**Bool PuedeJugar(Ficha ficha).**
Descripción	Para saber si una ficha puede ser puesta en el estado actual. Por cualquiera de las cabezas.
Nombre	**Bool PuedeJugar(Ficha ficha, bool cabeza).**
Descripción	Para saber si una ficha puede ser puesta en el estado actual por la cabeza especificada.

Anexo 5

Descripción de la Clase: Accion

Nombre	Accion
Tipo de clase	Entidad
Resumen	Esta clase tiene la responsabilidad de guardar los datos del resultado de una acción.
Atributo	**Tipo**
tipo	TipoAccion
jugador	int
llamada	ResultadoLlamada

Anexo 6

Descripción de la Clase: Historial

Nombre	Historial
Tipo de clase	Entidad
Resumen	Esta clase guarda el conjunto de acciones de un juego completo.
Atributo	**Tipo**
cantidadJugadores	integer
cantidadAcciones	integer

Anexo 7

Descripción de la Aplicación Gráfica: SecuenciaFichas

Nombre	SecuenciaFichas
Tipo de clase	Controladora
Resumen	Esta clase tiene el objetivo de controlar la secuencia de fichas en una ronda.
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre	**void AgregarFicha(Ficha ficha, PosicionCabeza lado)**
Descripción	Agregar una ficha por la cabeza especificada.
Nombre	**void EliminarFicha(Ficha ficha)**
Descripción	Eliminar una ficha de la secuencia.
Nombre	**Ficha Inicio()**
Descripción	Obtener la ficha inicial
Nombre	**Ficha[ ] SecuenciaDerecha()**
Descripción	Obtener la secuencia derecha a partir de la ficha inicial.
Nombre	**Ficha[ ] SecuenciaIzquierda()**
Descripción	Obtener la secuencia izquierda a partir de la ficha inicial.

Nombre	**Orientacion OrientacionInicio()**
Descripción	Para obtener como esta orientada la ficha inicial.
Nombre	**void AgregarContinuacion(continuacion : Continuacion)**
Descripción	Agregar una continuación a la secuencia de jugadas.
Nombre	**Void Reiniciar()**
Descripción	Reiniciar la secuencia. Vacia la lista y pone la secuencia lista para empezar otra ronda.

Anexo 8

Descripción de la Aplicación Gráfica: EstadoJuego

Nombre	EstadoJuego	
Tipo de clase	Controladora	
Resumen	Esta clase tiene el objetivo de controlar todo el estado del juego, traducir y llevar a cabo las acciones que ocurran.	
Atributo		**Tipo**
Para cada responsabilidad:		
Nombre	**void EjecutarAccion(Accion accion)**	
Descripción	Ejecutar una accion especifica en el sistema	
Nombre	**SecuanciaFichas Secuencia()**	
Descripción	Obtener la secuencia actual de fichas en el tablero.	
Nombre	**ConjuntoFichas[ ] Conjuntos()**	
Descripción	Obtener el conjunto de fichas que tiene cada jugador.	

Anexo 9

Descripción de la Aplicación Gráfica: Draw

Nombre	Draw	
Tipo de clase	Controladora	
Resumen	Esta clase tiene el objetivo de pintar los diferentes elementos del juego.	
Atributo		**Tipo**
Para cada responsabilidad:		
Nombre	**void PintarSecuenciaFichas(SecuenciaFichas secuencia, Graphics g)**	
Descripción	Pintar una secuencia de fichas al estilo cotidiano de juego.	
Nombre	**void PintarFicha(Ficha ficha, Point centro, Orientacion orientacion, Graphics g)**	
Descripción	Pinta una ficha en la posición y con la orientacion especificada.	
Nombre	**void PintarConuntoFicha(ConjuntoFichas conjunto,LadoJugador lado, Graphics g)**	
Descripción	Pintar el conjunto de fichas al lado del tablero especificado.	
Nombre	**void PintarTablero()**	
Descripción	Pintar el tablero.	

Glosario de términos

3D Studio MAX: 3D Studio Max es una aplicación basada en el entorno Windows (9x/NT) que permite crear tanto modelados, como animaciones en tres dimensiones (3D) a partir de una serie de vistas o visores (planta y alzados). La utilización de 3D Studio Max permite al usuario la fácil visualización y representación de los modelos, así como su exportación y salvado en otros formatos distintos del que utiliza el propio programa.

EverQuest: Es un famoso juego para múltiples jugadores de forma online. El juego posee un rico entorno en 3D y su extensión es tan grande que muy pocos jugadores han estado en las 230 zonas que conforman el mundo virtual del juego.

Asheron's: Es un popular juego que participan diversos jugadores de forma online, en el mismo los jugadores podrán vivir incontables aventuras en un vasto mundo donde la magia y la lucha son una constante.

World of Warcraft: Comúnmente conocido por las siglas WOW, es un videojuego para múltiples jugadores de forma online desarrollado por la compañía Blizzard Entertainment, disponible para los sistemas operativos Windows y Mac OS X.

WWW (World Wide Web): es un sistema de documentos de hipertexto y/o hipermedios enlazados y accesibles a través de Internet.

DLL: (Dinamyc Link Library). Es una librería de código binario que puede ser cargada por cualquier lenguaje en tiempo de ejecución.

Common Language Runtime (CLR): es el núcleo de la plataforma .NET. Es el motor encargado de gestionar la ejecución de las aplicaciones para ella desarrolladas y a las que ofrece numerosos servicios que simplifican su desarrollo y favorecen su fiabilidad y seguridad.

Integrated Development Environment (IDE): es un programa compuesto por un conjunto de herramientas para un programador.

.NET: es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

API (Application Programming Interface): es un conjunto de funciones residentes en bibliotecas (generalmente dinámicas) que permiten que una aplicación corra bajo el sistema operativo Windows.

Software Development Kit (SDK) o kit de desarrollo de software: es un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema bastante concreto, por ejemplo ciertos paquetes de software, frameworks, plataformas de hardware, ordenadores, videoconsolas, sistemas operativos, etcétera.

Internet: interconexión de redes informáticas que permite a los ordenadores o computadoras conectadas comunicarse directamente.

Inteligencia artificial (IA): término que, en su sentido más amplio, indicaría la capacidad de un artefacto de realizar los mismos tipos de funciones que caracterizan al pensamiento humano.

SQL (Structured Query Language): es un estándar aceptado en productos de bases de datos, se utiliza en el diseño de consultas interactivas y puede incluirse en una aplicación como un conjunto de instrucciones de manejo de datos (sentencias).

UML (Lenguaje Unificado de Modelación): es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos del sistema de un Software.