Facultad 5

Módulo Lógico para el juego de simulación Indicios Aduaneros

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autoras: Irais Aldana Llanes.

Yaimara Ochoa Santiesteban.

Tutores: Ing. Jaime González Campistruz.

Ing. Lien Muguercia Torres.

Ciudad de la Habana, Cuba.

No tenía miedo a las dificultades: lo que asustaba era la obligación de tener que escoger un camino.

Escoger un camino significaba abandonar otros.

Paulo Coelho

Datos de contacto

Tutor:

Ing. Jaime González Campistruz.
Universidad de las Ciencias Informáticas, La Habana, Cuba.
e-mail: jgonzalezc@uci.cu.
Teléf.: 3173.
Apto: 131104.

Co-Tutora:

Ing. Lien Muguercia Torres.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

e-mail: lmuguercia@uci.cu.

Teléf.:

Apto:

Autoras:

Irais Aldana Llanes.
Universidad de las Ciencias Informáticas, La Habana, Cuba.
e-mail: ialdana@estudiantes.uci.cu.
Teléf.: 3085.
Apto: 93203.

Yaimara Ochoa Santiesteban.

Universidad de las Ciencias Informáticas, La Habana, Cuba.
e-mail: yochoa@estudiantes.uci.cu.
Teléf.: 3085.
Apto: 93203.

Agradecimientos

Quisiera agradecer primeramente a mi madre por haberme dado la vida y dedicarme la suya sin esperar nada a cambio, por siempre estar cuando la necesité y apoyarme en todas mis decisiones incluso cuando no estaba de acuerdo con mi forma de actuar, a mi padre por sus consejos y confianza en mí. A mi tía Zoila, tío Héctor y mis primas Yadira y Yamilka por su apoyo incondicional en estos 5 años, por haber hecho mi estancia en la UCI más placentera, principalmente los fines de semana con su gran sentido del humor que tanto me gusta. A mis abuelos Tete y Felipe que siempre estuvieron para ayudarme a salir adelante, a mi novio por estar a mi lado siempre y quererme tanto, por sus ocurrencias que hicieron que los últimos 4 años fueran los mejores. A todos mis amigos de la universidad que son amigos para toda la vida, nunca los voy a olvidar, principalmente a las dinémicas (Ira, Yai, Lisby) por tantos momentos lindos que compartimos. También a mi amiga y compañera de tesis Irais por su dedicación y soportar mis malacrianzas. A nuestro tutor Jaime por tener tanta paciencia con nosotras.

Yaimara

Quisiera agradecer en primer lugar a mis padres, las personas más importantes de mi vida y sin los cuales nada de esto fuera posible. A mi madre por poner en segundo plano su vida desde que comenzó la mía, por su grandísimo esfuerzo y dedicación, por dar cada gota de su sudor porque yo estuviese lo mejor posible sin importar qué quedara para ella, por su manera tan linda de amarme y cuidarme. A mi padre por estar siempre presente en cada momento, por ser el mejor padre del mundo, por acompañar y cuidar a mi madre en cada uno de sus días, por depositar esa confianza en mi cada vez que decía, "yo confío más en ti niña que en los varones", por ser el que me da fuerzas en los momentos malos, por quererme tanto. A mis hermanos que siempre han estado ahí para mi, por cuidarme como su tesoro más preciado, por guiarme siempre por el mejor de los caminos, a ti Alvaro por entenderme aún cuando ni yo misma lo hacía. A Emirelis por haber estado presente estos cinco años sin ninguna obligación, por todo el amor, la dedicación y la preocupación que has tenido conmigo, te quiero mucho. A mis abuelos Mamá y Tío por ser mis segundos padres, y a Papá y Mamá por siempre desear lo mejor para mí. A cada miembro de mi familia por esa manera especial de

quererme. A mi mejor y grandísima amiga, mi compañera de tesis, la que ha sabido entenderme, cuidarme y estar incluso en momentos que ni siquiera lo necesitaba pero simplemente yo quería, por su paciencia y por la gran persona que es, gracias Yaimi por aconsejarme y regañarme de la forma más amable que he visto jamás. A mi familia de la UCI, los amigos que siempre han estado ahí para mí en cada momento, y que es imposible dejar de mencionar. A todos ustedes, mi hermanita Leinys te quiero muchísimo y lo sabes, Yaima, Lisbey, Yoisell, Suset, Yeisnier, Yaniel, Roly. A mi tutor por su paciencia, dedicación, por ser la gran persona que es, por su ayuda en cada momento. A cada uno de los miembros de mi aula que durante estos 5 años hemos pasado miles de momentos buenos y malos. A todos los amigos que he tenido la oportunidad de conocer muchísimas gracias. Al destino por darme la oportunidad de haber vivido estos 5 años, aunque lejos de la gente que más amo, pero gracias porque he aprendido además de física y programación, a tomar mi propias decisiones, malas o buenas pero al final mías, porque aprendí a creer en mí hasta en los peores momento, a escoger mi propio camino, por convertirme en la mujer que hoy soy.

Irais

Dedicatoria

Este trabajo está dedicado a toda mi familia en especial a mi madre que fue quien logro que llegara hasta el final, también se la dedico a mi papá que se esto lo hace muy feliz y a mi prima Milena que se guía mucho por mí, quisiera que llegaras más lejos que yo.

Yaimara

A mis padres por darme la vida y por hacérmela cada día mejor, a mis hermanos por cuidarme tanto y a mi sobrina Claudia para que un día escoja un camino mucho mejor que este.

Irais

Resumen

Hoy en día en la Aduana General de la República (AGR) no existe un sistema automatizado que permita a los agentes aduaneros realizar sus entrenamientos de forma más económica y segura. Que contribuya a la formación de este personal para la detección del arribo de personas en los aeropuertos del país con tráfico de drogas, armas, elementos terroristas y otros, accidentes que deben ser minimizados pues impactan de forma negativa en la sociedad. Es por eso que en coordinación con la Facultad 5 de la Universidad de Ciencias Informáticas (UCI), la AGR solicita el desarrollo de un sistema automatizado que permita realizar dichas prácticas. Para el funcionamiento correcto de este sistema se necesita de un módulo lógico, el cual será la propuesta de este trabajo.

Palabras claves:

Módulo, lógica, videojuego, diagrama, juegos serios.

Índice

Introducción

La Aduana General de la República es la división responsable de la seguridad nacional en los diferentes puertos y aeropuertos del país. De esta organización depende en gran medida, anular el tráfico de personas, drogas, armas, ataques terroristas, entre otras ilegalidades con impacto negativo en la sociedad.

La preparación del personal es de vital importancia para asegurar que en un futuro se cuente con agentes aduaneros altamente capacitados en la mayor medida posible. Actualmente el proceso de entrenamiento se realiza fundamentalmente mediante presentaciones y test elaborados por especialistas de la Escuela Nacional de Formación Aduanera (ENFA) y en ocasiones, estos inspectores son llevados a los salones de operaciones de la aduana en los aeropuertos, a poner en práctica sus conocimientos.

El uso del primero de estos medios de enseñanza es bastante ineficiente pues limita a los educandos a observar solo situaciones predefinidas que en algún momento llegan a aprender de memoria y dejan de aportar las experiencias necesarias.

El segundo método expuesto tributa los conocimientos que necesitan los futuros aduaneros, pues se familiarizan con el ambiente adecuado y las situaciones cambian con mayor aleatoriedad por desarrollarse en la propia realidad. Sin embargo, tiene la desventaja de que en la práctica directa con los pasajeros que arriban a los aeropuertos ocurra algún incidente, como la entrada al país de personas con documentación falsificada, armas, drogas, elementos terroristas y otros.

La AGR no cuenta con un sistema automatizado que permita a los agentes aduaneros recién graduados realizar sus entrenamientos de forma más económica y segura, ayudándolos a adquirir experiencias sin correr los riesgos antes mencionados y asegurándose de brindarle la mayor cantidad de adiestramiento posible a la hora de detectar algún posible lugar de ocultamiento de artefactos ilícitos.

Es por eso que en coordinación con la Facultad 5 de la Universidad de Ciencias Informáticas, encargada de los procesos de Realidad Virtual (RV), la Aduana solicita el desarrollo de una aplicación para mejorar la preparación del personal aduanero. Para el funcionamiento correcto de este sistema se necesita de un Módulo Lógico, que será el encargado de las reglas del juego, y todo lo referente a la lógica de las respuestas que da el sistema a la aplicación. Este será la propuesta de la presente tesis.

Teniendo en cuenta las necesidades y la situación de la AGR se plantea como **Problema científico** a solucionar ¿Cómo garantizar el cumplimiento de las acciones que se realizan en la Aduana para la detección de indicios, a partir de su simulación como medio de entrenamiento eficaz para el personal aduanero?

Este trabajo tiene como **objeto de estudio** los juegos de simulación y se tendrá como **Campo de acción** módulo lógico para juegos de simulación.

Como **Objetivo general del trabajo** se realizará un módulo lógico para el juego serio Indicios Aduaneros.

Para alcanzar los objetivos propuestos se realizarán las siguientes **Tareas investigativas**:

- ✓ Estudio de las tendencias actuales en la creación de juegos de simulación, para la identificación de las mejores técnicas usadas en la elaboración de este tipo de juegos.
- ✓ Caracterización del proceso de detección de indicios llevado a cabo por el personal de la aduana, para precisar la simulación correcta.
- ✓ Análisis de la arquitectura del juego de simulación Indicios Aduaneros, para acoplar el módulo lógico de manera efectiva.
- ✓ Desarrollo del Modelo del Negocio, para identificar los requisitos y los Casos de Uso (CU).
- ✓ Desarrollo del Modelo del Sistema, para identificar los requisitos y los CU.
- ✓ Desarrollo de la etapa de Análisis y Diseño, para la creación de los diagramas necesarios para la implementación del módulo.
- ✓ Desarrollo del flujo de Implementación.
- ✓ Realización de la fundamentación teórica.
- ✓ Implementar una parte del módulo lógico del proyecto Indicios Aduaneros.

Capítulo 1: Fundamentación teórica

Actualmente los videojuegos han tomado gran auge a nivel mundial. Ejemplo claro de esto es que el mercado de los videojuegos mueve anualmente 30.000 millones de dólares al año (phpBB SEO, 2007). Esta rama del entretenimiento frente a una computadora tiene varios géneros, entre los que se destacan Aventura, Estrategia, Lucha, Plataformas, Videojuegos de rol, Musicales, Deportivo, Carreras. Pero este trabajo se centrará específicamente en el área de los juegos serios. Un término surgido hace muy poco y que recoge a aquellos juegos que tienen como objetivo central transmitir algún tipo de conocimiento.

Para el funcionamiento correcto de un videojuego son necesarios varios módulos, que dividen y organizan la creación de éste, facilitando el trabajo de los desarrolladores. Entre los más importantes se destacan Módulo Gráfico, Módulo de Red, Módulo de Sonido; así como el Módulo Lógico, que será en el que se enmarcará la actual investigación.

Generalmente cuando se habla de la lógica del juego se piensa solamente en la implementación de la inteligencia de los personajes del videojuego. Es importante puntualizar que se tienen que programar todos los comportamientos de las escenas y las reglas que rigen cada nivel.

En este capítulo se expondrán los conceptos fundamentales de Videojuegos, Juegos Serios, Módulo o Motor para un juego, así como la explicación más detallada de qué es y cuáles son las principales funciones del Módulo Lógico.

1.1 Definición de Videojuegos

“Un videojuego, del inglés video-game, se conoce como un programa de computación, creado para el entretenimiento, basado en la interacción entre una o varias personas y un aparato electrónico, ya sea un ordenador, un sistema arcade, una videoconsola, un dispositivo handheld o actualmente un teléfono celular, el cual ejecuta el videojuego. En muchos casos, estos recrean entornos y situaciones virtuales en los cuales el jugador puede controlar a uno o varios personajes o cualquier otro elemento de dicho entorno, para conseguir un determinado objetivo siguiendo determinadas reglas” (Departamento de Comunicaciones Comfamiliar Tuluá, 2009).

Según el diccionario de la Real Academia Española, videojuego es un: "dispositivo electrónico que permite, mediante mandos apropiados, simular juegos en las pantallas de un televisor o de un ordenador" (Indice Latino, 2007).

Pero más específicamente se definen a los videojuegos como:

“Videojuego es aquel programa informático, normalmente asociado a un hardware específico, que recrea un ejercicio sometido a reglas, se debe lograr uno o varios objetivos, donde los jugadores pueden interactuar y tomar decisiones” (Duch i Gavaldà, et al., 2005).

1.2 Surgimiento de los videojuegos

Desde su surgimiento hasta la actualidad los videojuegos han pasado por una gran transformación y avance. De los que han sido protagonistas compañías líderes como Nintendo, Atari y SEGA, entre muchas otras.

Algunas bibliografías reflejan a el *Nought and crosses* u OXO, creado por Alexander S. Douglas en el 1952, como el primer videojuego. Este no era más que una imitación del ¨tres en raya¨, se ejecutaba sobre la *Electronic Delay Storage Automatic Calculator* (EDSAC) y podían enfrentarse el jugador con la computadora. También se ha hablado mucho de William Higinbotham, quien fue un destacado físico de Estados Unidos, participante en el desarrollo de la bomba atómica. En una de sus exposiciones en 1958, tratando de entretener a sus visitantes diseñó un sistema de juego que simulaba al tenis en el cual con una línea horizontal, y otra pequeña céntrica vertical, a modo de red, simulaba una pista de tenis. Este invento de Higinbotham no tenia fines comerciales, solo se vio como una curiosidad científica. Después de unos 15 años la idea se perfeccionó y sumándole algunos cambios y un marcador para llevar la puntuación, se convertía entonces en el juego más popular de aquel entonces. En la nueva versión aparecían dos líneas paralelas a modo de raqueta, una céntrica que dividía la pantalla en dos bloques y un gran pixel que funcionaba como bola en una breve simulación de tenis de mesa. Atari se encargó de desarrollarlo y comercializarlo bajo el nombre de: Pong, que apareció en las primeras maquinas arcade recreativas.

En realidad un año antes Nolan Bushnel, creador de Atari, había elaborado el primer arcade; que es el término genérico que identifica a las grandes máquinas de videojuegos disponibles usualmente en lugares públicos de diversión, centros comerciales, restaurantes, bares, o salones recreativos especializados. En aquellos momentos podían participar 1 o 2 jugadores y lo llamaron "*Computer Space*".

Como sucesión a estos hechos siguieron una serie de acontecimientos, todos movidos por la competencia y encaminados al desarrollo y perfeccionamiento de cada detalle de los videojuegos que salían al mercado. Así han evolucionado los videojuegos desde aquel sencillo y arcaico juego de Ping Pong, que apenas contaba con opciones, hasta los actuales FIFA o Warcraft que desarrollan gráficos excelentes, con posibilidad de jugar en red y con un guión que es capaz de mantener a los usuarios frente al monitor por horas.

La industria del videojuego surgió con fuerza en Estados Unidos, Europa y Australia con la llegada del tenis de mesa y otros juegos muy sencillos a finales de la década de 1970 y se extendió rápidamente por todo el mundo. Tras conocer un crecimiento espectacular a lo largo de la década de 1980, la industria japonesa, especialmente la *Nintendo Company Ltd*, se lanzó de lleno a perfeccionar y desarrollar la tecnología del juego, introduciendo juegos tan populares como el Super Mario Bros. Las empresas afincadas en Japón, como Nintendo y SEGA, continúan dominando el mercado mundial.

Desde 1993 estas dos compañías están realizando esfuerzos para controlar y establecer el contenido de los juegos. La iniciativa responde a las críticas, especialmente de los padres, preocupados por la intensificación de la violencia y la introducción de temas para adultos en los juegos infantiles.

1.3 Clasificación de los videojuegos

Hoy en día se ha creado una innumerable cantidad de juegos computarizados, entre las características fundamentales se destaca el gran avance que se alcanzado en cuanto a resultado final, credibilidad y estimulación que logran trasmitir a los usuarios. Algunos defensores de los juegos afirman que entre otras muchas cosas, ayudan a los niños y adolescentes a adentrarse en el mundo de la informática, así como interesarse por él. Pues está demostrado de sobra que actualmente se ha globalizado el uso de las computadoras en todos los sectores de la sociedad, y prácticamente ya es imposible desligar cualquier tarea de la digitalización, o por lo menos cada sector de la sociedad tiene al menos un matiz de la Informática.

Con toda la expansión que han alcanzado los ya muy mencionados videojuegos, los expertos en el tema se han encargado de clasificarlos en diferentes géneros. Entre los más destacados están (Duch i Gavaldà, et al., 2005):

- ✓ Aventura: Juegos en los que el protagonista debe avanzar en la trama interactuando con diversos personajes y objetos.

- ✓ Deportivos: Se basan en deportes, reales o ficticios.
- ✓ Disparo: En estos juegos el protagonista ha de abrirse camino a base de disparos.
- ✓ Educativos: Juegos cuyo objetivo dar a conocer al usuario algún tipo de conocimiento.
- ✓ Lucha: Juegos basados en el combate cuerpo a cuerpo.
- ✓ Rol: Se inspiran en los juegos de rol clásicos, donde el protagonista interpreta un papel y ha de mejorar sus habilidades mientras interactúa con el entorno y otros personajes.
- ✓ Simulación: Involucran al jugador en una situación simulada determinada, ya sea de gestor de un zoo, una ciudad o una vida propia virtual.
- ✓ Estrategia: Se caracterizan por la necesidad de manipular a un numeroso grupo de personajes, objetos o datos para lograr los objetivos.
- ✓ Carreras: Son juegos en los que se pilotan diferentes vehículos, ya sean reales o ficticios, para ganar en diferentes carreras.
- ✓ Horror: Correspondientes al género de terror.
- ✓ Plataformas: Los juegos de plataformas o, simplemente, plataformas, son un género de videojuegos que se caracterizan por tener que andar, saltar o escalar sobre una serie de plataformas.
- ✓ Musicales: Su desarrollo gira en torno a la música.

Roger Caillois, escritor francés, antropólogo y ensayista, los clasifica también en:

- ✓ *Agon*/Competición: Juegos de competición como deportes, dardos.
- ✓ *Alea*/Suerte: Juegos de azar como casino.
- ✓ *Mimicry*/Simulación: Juegos de simulación de una realidad que incluso puede ser ficticia, por ejemplo juegos de disfraces, mímica.
- ✓ *Ilinx*/Vértigo: Juegos que buscan el desequilibrio del cuerpo, un trance o aturdirse momentáneamente. Por ejemplo, tirarse rodando por una ladera.

Gonzalo Frasca, uno de los referentes en el estudio teórico de los videojuegos. En el año 2001 creó www.ludology.org, un *weblog* que sirve de punto de encuentro para aquellos interesados en la ludología, es decir, el estudio de los videojuegos desde una perspectiva más humana y social. Este destacado estudioso de los videojuegos argumenta que la anterior calcificación tiene muchos solapamientos entre los grupos, entonces plantea la siguiente alternativa:

- ✓ *Ludus*: Juego donde las reglas son rígidas.

- ✓ *Paidea*: Juego donde la acción es libre.

A medida que los videojuegos se han ido desarrollando, es común que uno de ellos pertenezca a más de un género y que cualquier clasificación que se haga de él deba considerarse como una aproximación. Por otro lado, los géneros tradicionales deben evolucionar de la misma manera que lo hacen los videojuegos, y es muy difícil mantener esa clasificación de manera coherente a medida que salen al mercado nuevos títulos con nuevos argumentos y formas de juego. También existen géneros que caen en desuso y sólo se utilizan para clasificar juegos antiguos.

1.4 Definición de Videojuegos Serios

El Juego Indicios Aduaneros (JIA) simulará el proceso llevado a cabo por el personal de la Aduana desde que arriba un avión a un aeropuerto cubano hasta que todos los pasajeros son observados, y registrados si es necesario, en cada uno de los cuatro salones por los que deben pasar, Finger, Salón de Inmigración, Pit de entrada y Salón de la Aduana, que serán entonces los niveles de la aplicación. El juego en cuestión tiene fines educativos, su objetivo principal es ayudar a los estudiantes, futuros agentes del personal de la AGR, a detectar cualquier infiltración en el país ya sea de terroristas, drogas o cualquier otra, todo mediante un sistema digital.

Este sistema debe ayudar a los estudiantes a acumular un grupo de experiencias y habilidades para detectar infiltraciones delictivas en el país. De esa forma se evita que por desconocimientos de algunos hechos, que pueden ser comunes para los agentes expertos, dichos educandos admitan la penetración al país de alguna de las amenazas antes mencionadas al realizar las prácticas de forma directa en los aeropuertos cubanos. Dada la explicación anterior se decide introducir el juego de simulación Indicios Aduaneros dentro del concepto de juegos serios, una clasificación surgida en el 2006 y que recoge a todos los juegos que su principal propósito es llevar a los usuarios algún conocimiento.

Una simple explicación que muchos profesionales utilizan en este campo, con algunas reservas y calificaciones, es: “Un juego serio es en el que la educación, en sus diversas formas es el principal objetivo, en lugar del entretenimiento” (Michael, et al., 2006).

La mayoría de los juegos son presentados a los jugadores potenciales como un entretenimiento agradable y una manera de pasar el tiempo o interactuar con otros jugadores.

Sin embargo se olvida que los juegos en cualquiera de sus formas pueden ser utilizados para la educación en todos los niveles, desde preescolar y escuela primaria, en colegios y universidades, e incluso en el mercado de trabajo. Un juego no tiene que apoyar a todos los niveles, pero a algunos podrían ser capaces de hacerlo.

La definición más simple de los juegos serios sería:

“Son los juegos que no tienen el entretenimiento, el disfrute o la diversión, como principales propósito. Esto no quiere decir que los juegos en virtud de la grave general no son juegos de entretenimiento, o divertidos. Es simplemente que hay otro objetivo, un motivo ulterior en un sentido real” (Michael, et al., 2006).

A continuación se muestra una encuesta expuesta en el libro *Serious Games* que refleja a quien van dirigidos principalmente los juegos serios, lo que reafirma lo planteado en el concepto de juego serio a cerca de que su objetivo principal es transmitir algún tipo de conocimiento lejos de divertir o entretener.

Pregunta: ¿quién ha sido el público objetivo los proyectos de juegos serios? (Michael, et al., 2006)

53,97% estudiantes (cualquier nivel).
47,62% público en general.
26,98% de gestión empresarial y / o ejecutivos.
23,40% los profesionales de la educación.
23,81% del personal de la administración.
22,22% empleados.
17,46% del personal militar.
7.94% pacientes de salud (incluyendo salud mental).
7.94% del personal médico de emergencia.
1,59% activista.
4,76% otros.
(Nota: 63 encuestados, en la encuesta se podía seleccionar más de una opción)

1.5 Idea general de la creación de un videojuego

Algunas bibliografías plantean que los primeros aspectos a tener en cuenta a la hora de comenzar con la creación de un videojuego son (Duch i Gavaldà, et al., 2005):

- ✓ Argumentación y discurso narrativo.
- ✓ Situación de la acción y su ambientación.
- ✓ Desarrollo de la acción: objetivos, partes.
- ✓ Derechos de explotación.

Estos aspectos agrupan la idea central y el prototipo del juego que se creará en un futuro. Pero es importante señalar que no necesariamente la idea del juego debe ser de los desarrolladores, algunas veces, como es este el caso, la petición, y con ella todos los requisitos, vienen de una empresa externa que es la que solicita el producto, que puede ser con diferentes fines, como por ejemplo a partir de una película, un libro o como en el caso del JIA con el objetivos del aprendizaje específicamente.

El próximo paso es crear el equipo de trabajo que va a desarrollar el videojuego. Generalmente se conforma con los siguientes integrantes:

- ✓ Jefe de Proyecto.
- ✓ Diseñadores.
- ✓ Grafistas.
- ✓ Programadores.
- ✓ Sonido.
- ✓ Producción.
- ✓ Testeo.
- ✓ Departamentos tradicionales de la empresa.

Después de tener bien definida la idea del juego que se desea construir, entonces se organiza el trabajo y se le asigna a cada uno de los integrantes del equipo de desarrolladores la parte que le corresponde.

Como la creación de un videojuego suele ser muy compleja y consta de muchísimos requerimientos entonces se divide en partes que se le denominan Módulos.

”Un módulo de un juego es el encargado de dirigir y coordinar cada uno de los aspectos tecnológicos ligados a él. Por ejemplo capta eventos de entrada, computa física (colisiones, proyectiles), reproduce sonido, simula Inteligencia Artificial (IA), pinta, etc.” (Duch i Gavaldà, 2007)

Muchos autores de la materia señalan que los módulos necesarios para el correcto funcionamiento de un videojuego son los que se exponen a continuación (Duch i Gavaldà, et al., 2005).

1.5.1 Módulo Físico

Este módulo es el encargado de la detención y reacción a colisiones y la simulación del movimiento de aquellas entidades que puedan ser sometidas a fuerzas externas.

La detección de las colisiones suele realizarse en dos niveles:

En primer lugar el mundo está divido por descomposición espacial, mediante el cual se ubican y mantienen actualizadas las posiciones donde cada elemento está presente dentro de todo el escenario. Algunos algoritmos que pueden ser utilizados con este objetivo son:

- ✓ BSP (*Binary Space Parttition*).
- ✓ *Quadttree/Octree*.
- ✓ *K-d tree.*

En un segundo término tiene lugar la detención de colisiones confrontando las entidades en cuestión entre sí, haciendo uso de volúmenes envolventes. Las estructuras más comunes utilizadas a tal efecto son:

- ✓ AABB (*Axis Aligned Bounding Boxes*).
- ✓ OBB (O*rieneted Bound Boxes*).
- ✓ *Bounding Sphere*.

1.5.2 Módulo de Red

Este módulo tiene una marcada presencia en los videojuegos hoy en día. La mayoría de estos ofrecen el servicio de comunicación que se produce entre diferentes ordenadores o videoconsolas a través de una red. Usando esta conexión, podemos interactuar con otros jugadores y participar de un modo cooperativo o competitivo en un Videojuego. Así se añade mayor entretenimiento, incluso para algunos es un medio imprescindible.

La comunicación entre dos dispositivos conectados mediante una red se realiza a través de lo que conocemos como un protocolo de comunicación, es decir, un conjunto de normas que definen la "lengua" en la que hablan dos ordenadores entre sí. El conjunto de protocolos más extendido en la actualidad y el que se utiliza actualmente en Internet es conocido por TCP/IP.

Es importante señalar además que en éste ámbito según las necesidades específicas se tratan las arquitecturas cliente-servidor, P2P, granjas grids o híbridas.

1.5.3 Módulo Gráfico

El motor gráfico es la parte de un programa que controla, gestiona y actualiza los gráficos 3D en tiempo real. Actualmente, todos los motores gráficos que existen utilizan las mismas técnicas (radiosidad, *mapping*/cartografía). Es el módulo más importante por su complejidad y la cantidad de puntos que trata, entre los que están:

- ✓ Almacenamiento de objetos, la técnica utilizada para tratar y guarda la información geométrica, vértices.
- ✓ Tipos de formatos para escenario o modelos: 3Ds, max, obj, mdl, md2, md3, etc.
- ✓ Técnicas para recrear el cielo envolvente: SkyBox, SkyPlane, SkyDone.
- ✓ Técnicas de visibilidad: portal cullin, BSP, octree, etc.
- ✓ Técnicas de iluminación y otras.

1.5.4 Módulo Sonido

Este es el módulo encargado de reproducir la música y los efectos de sonido de un videojuego. En este subsistema, se ofrece la reproducción de varios formatos, como mp3, wav, obb, midi y otros. La técnica de sonido 3D se brinda para dar una mayor inmersión, escuchar aquellos efectos que nos son cercanos con mayor intensidad y por el altavoz adecuado.

1.5.5 Módulo Inteligencia Artificial

El aumento del rendimiento de las tarjetas gráficas de los últimos años ha hecho que se haya incrementado el porcentaje dedicado a los cálculos de comportamiento y demás aspectos de la IA. Precisamente se pueden dividir los aspectos ligados a ésta en dos grandes grupos.

Las técnicas robustas:

- ✓ FSM (Finite State Machine).
- ✓ Sistemas de reglas.

- ✓ *Pathfining*: Búsqueda de ruta optima.
- ✓ Dinámicas de grupos.
- ✓ *Scripting*.

Las técnicas experimentales:

- ✓ Sistemas de aprendizaje.
- ✓ Algoritmos genéticos.
- ✓ Redes neuronales.

1.6 Módulo Lógico de un videojuego

Después de haberse hecho un breve estudio de las principales partes por las que está compuesto un videojuego, en este epígrafe se abordará a cerca del Módulo Lógico que será el resultado final del trabajo. Generalmente se tiene la idea que la lógica del juego es solamente la implementación de la inteligencia de los personajes del videojuego. También se tienen que programar los comportamientos de todas las escenas, por ejemplo, "si hieren a un personaje, que sangre".

"La parte donde se controla el desarrollo del juego es lo que se denomina motor de lógica. Esta parte, tan o más importante que todas las descritas anteriormente, incluye la descripción de los atributos de todos los elementos que participan, y de todas las reglas y condiciones que se sitúan en el juego. Continuamente chequea las acciones que han realizado los jugadores y los elementos controlados por la IA y decide si estas acciones se pueden llevar a cabo y cuál es el resultado de ejecutarlas" (Duch i Gavaldà, 2007).

En la lógica del juego se diseñan y programan todos los algoritmos necesarios para definir e implementar las reglas del juego, la interacción entre los elementos y los comportamientos de todos los avatares. Es el gran núcleo del videojuego y donde más programadores son necesarios. En algunos libros relacionados con el tema se define un juego como un conjunto de objetivos que se deben conseguir siguiendo una serie de reglas. La implementación de estas reglas, y por tanto del juego, se realiza en lo que se denomina el motor lógico de un videojuego, también llamado motor de lógica. Tecnológicamente hablando, el motor lógico es el componente más importante de un videojuego y tiene entre sus funciones (Duch i Gavaldà, 2007):

- ✓ Inicializar el contenido del juego y crear el mundo donde se desarrollará la acción.

- ✓ Determinar cuándo empieza y cuándo termina una partida.
- ✓ Implementar las reglas que controlan las acciones de los jugadores y controlar que estas reglas se cumplen durante el desarrollo de la partida.
- ✓ Permitir que los jugadores interactúen con el entorno sin romper estas reglas.
- ✓ Controlar el estado de todos los elementos que están participando en la partida, desde las posiciones de los avatares hasta las variables globales de la partida: tiempo de juego, puntuación, resultados.

Actualmente una de las tendencias que se sigue para representar el motor lógico es utilizar una descripción particular del mundo en el que se esté desarrollando la acción. Por ejemplo a la hora de representar un personaje se obvian todas las características referentes al color, forma, animación u otras que no den una información precisa de la posición. Que se puede obtener entonces abstrayendo todo lo antes mencionado y centrándose en posiciones y vectores.

Esta simplificación es necesaria para que el ordenador pueda interpretar en todo momento el estado del juego. Por ejemplo para el sistema de IA no es necesario conocer si un personaje es de color verde o rojo para tomar una decisión de qué hacer con él, simplemente está interesado en el tamaño, la posición y la dirección en la que está mirando este orco.

Durante el desarrollo de un videojuego a veces es interesante poder observar el mundo lógico y el mundo físico/gráfico por separado. Esto permite trabajar con más detalle con uno u otro y depurar mejor los posibles problemas existentes en cada uno, de otra manera, se podría tener un exceso de información en pantalla.

1.6.1 Discretización del mundo

Una de las primeras tareas para definir el mundo lógico es realizar una discretización del mundo real que se va a utilizar. Dependiendo del tipo de juego y de los recursos del sistema se emplea una simplificación con más o menos detalle.

La discretización permite organizar con mayor eficiencia la distribución de los objetos en el mundo y asignar diferentes propiedades a los elementos que componen el mundo discreto.

Mediante la discretización también se puede crear un grafo que determine todos los posibles caminos entre dos puntos del mundo lógico y mediante algoritmos de búsqueda de caminos se pueden calcular las

rutas de los elementos móviles del sistema. Existen dos técnicas comunes para discretizar el mundo: sobreponer un grid regular encima del mundo o extraer una malla de navegación a partir de la geometría.

1.6.2 Grid regular

Un grid es una teselación de polígonos, que no es más que cubrir una superficie con un patrón de figuras planas que en este caso serian los polígonos y que estos no se superpongan ni dejen huecos. Cuando se sobrepone encima del mundo se puede determinar qué parte de este, se encuentra dentro o fuera del grid.

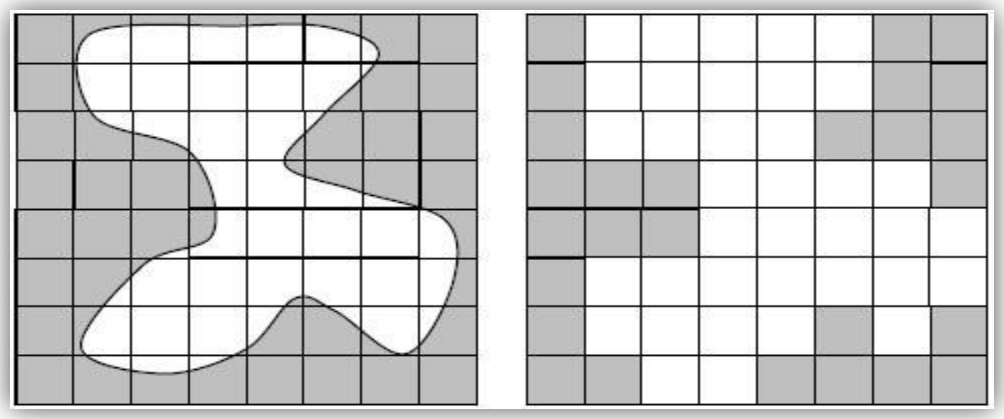

Figura 1: Ejemplo de un mapa con un grid rectangular (izquierda) y del resultado de la discretización (derecha). (Duch i Gavaldà, 2007)

Un grid regular solo puede estar compuesto por tres tipos de polígonos: triángulos equiláteros, cuadrados o hexágonos.

Cuanto más detalle se utilice para el grid, mayor será la resolución del sistema, pero también más espacio para guardarlo y mayor será el tiempo para calcular caminos sobre el mismo.

Para el cálculo de caminos se conectan con una línea el centro de todo par de polígonos adyacente. Esto dará el grafo de todas las rutas existentes en este grid. En el caso del grid cuadrangular es importante considerar si se puede ir directamente a las casillas que se encuentran en diagonal.

1.6.3 Malla de navegación

Una malla de navegación es una partición de la geometría del mundo lógico en polígonos convexos como los analizados anteriormente. Es un conjunto de polígonos que cubre por completo el mapa, donde cada par de polígonos solo comparte dos puntos y una línea. Cada polígono representa un punto de un camino que está.

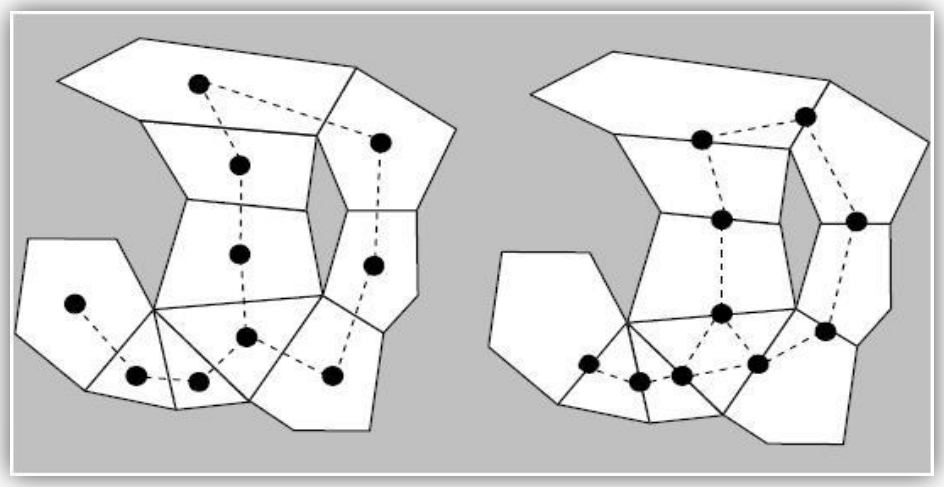

Figura 2: Ejemplo de una malla de navegación con los caminos usando los centros de los polígonos (izquierda) o las aristas (derecha). (Duch i Gavaldà, 2007)

1.7 Tareas del motor de lógica

El motor lógico se encarga de controlar todo el desarrollo del juego. Obtiene las acciones que han tomado los jugadores en un turno, de los sistemas de entrada ya sean los jugadores o la IA, y decide que sucede con ellas, calcula el nuevo estado global del juego y lo devuelve al sistema de lógica y a los dispositivos de salida para informar lo que ha ocurrido.

Entre las tareas más importantes que debe realizar el Motor de Lógica se encuentran las que se explican a continuación.

1.7.1 Aplicar movimiento de los elementos del juego

Para que sean cambiados los estados del juego, el motor lógico necesita que se le envíen las acciones que han tomado los componentes móviles o activos del juego en cada instante.

Una de las primeras tareas que se debe realizar en cada iteración de este motor es calcular las nuevas posiciones de todos los elementos. Existen dos posibles orígenes de movimiento:

1-Los elementos controlados por jugadores: La información de movimiento puede estar dada de varias maneras, posiciones relativas, absolutas, velocidad o aceleración. Esto depende de cómo se interprete la entrada del usuario y el estado actual del jugador. Por ejemplo, si se tiene un coche moviéndose con una cierta velocidad, aunque el usuario no realice ninguna acción en esta iteración se considera que el elemento se sigue moviendo.

2-Los elementos no controlados por jugadores: Estos movimientos los proporciona el sistema de IA. Algunos de ellos son movimientos cíclicos, con una ruta prefijada, y por tanto fácil de calcular. Otros movimientos son más complejos y requieren de la observación del estado del mundo para poder tomar decisiones. El sistema de IA puede enviar la información de varias maneras, así que en este caso también se tiene que interpretar el movimiento y traducirlo a posiciones reales para cada elemento.

Cuando se hayan calculado todas las posiciones, entonces se empiezan a calcular las colisiones entre los elementos. Aunque se haya introducido en el módulo de gráficos, la física y el control de colisiones lo realiza el motor lógico usando la información contenida en su representación del mundo y no por parte del motor gráfico. Tiene gran importancia decidir cuáles son los elementos con mayor prioridad respecto a otros al calcular las nuevas posiciones, ya sea utilizando sus propiedades físicas, por ejemplo el objeto con más masa y velocidad tendrá mayor prioridad, o utilizando prioridades que se asignen a los objetos. Después de haber pasado este filtro se comprueba por último que las posiciones finales de los elementos cumplan con las reglas del juego, es decir, que la posición final de cada elemento se encuentre dentro de una zona permitida para él.

1.7.2 Aplicar acciones de los elementos del juego

Otra tarea importante del motor de lógica es la de llevar a cabo todas las otras acciones que llevan a cabo los elementos descritos en el apartado anterior. La naturaleza de estas acciones puede ser muy variada, ya sea lanzar un hechizo, manejar un balón o construir una casa.

Primero se comprueba si se pueden realizar las acciones. Para esto se debe controlar que el elemento que realice la acción cuente con todo lo necesario para ejecutarla. Por ejemplo, en el caso del hechizo que el personaje tenga los atributos u objetos necesarios como energía, reactivos o en el caso del jugador que tenga la posesión del balón.

En segundo lugar, controlar que la acción se pueda desarrollar sin quebrantar las reglas del juego. Por ejemplo, aunque se tenga suficiente dinero para construir la casa, el sistema tiene que comprobar que se construye en una posición donde esté permitida por las reglas del juego.

Es importante que se decida quién realiza primero cada acción, sobretodo en el caso en que la acción afecte a otros elementos del juego. Una acción puede cambiar los atributos de otro elemento y evitar que este pueda llevar a cabo la suya. Por ejemplo, si dos jugadores se atacan entre sí, es posible que el primero acabe con el segundo y que éste ya no puede devolver el golpe. Por lo tanto, es muy importante que se decida la prioridad de las acciones de los elementos para no perjudicar a nadie.

Existen varias formas de decidir el orden de las acciones:

- ✓ Una forma es mediante la utilización de un sistema round-robin, en el que siempre se utiliza el mismo orden de procesamiento de acciones.
- ✓ Otra posibilidad es la selección del orden en que se llevan a cabo las acciones de forma aleatoria. En este caso solo se utiliza la aleatoriedad cuando las acciones se solapen, en otro caso se puede seguir con un round-robin.
- ✓ Finalmente, la opción más elegante es asignar prioridades a las acciones. Esta opción se puede complementar con las dos anteriores.

Una vez realizada cada acción se calculan los cambios del entorno debidos a la misma y se guardan en todos los elementos que se han visto afectados.

1.7.3 Cálculo del nuevo estado del juego

Por último se recalcula el estado global del sistema y actualizan las variables globales cuando todos los elementos hayan actuado y se hayan movido. Este paso se realiza al final para poder integrar el resultado de las acciones colectivas de los elementos que participan en esta iteración.

Algunas de las tareas que se deben realizar en este paso final son:

- ✓ Es necesario incrementar el tiempo del juego proporcionalmente. Por ejemplo, hay juegos donde es importante la distinción entre noche y día.
- ✓ En segundo lugar, se controla si el juego ha llegado a su fin. En este caso se tiene que informar al bucle principal que se tiene que cambiar de estado para dejar de procesar las entradas del usuario.

- ✓ Valorar el cumplimiento del objetivo intermedio, adoptando decisiones según la necesidad del cambio.

Toda esta información se deja almacenada en las estructuras de datos del sistema y se avisa a los componentes que lo necesiten como la IA, gráficos etc. de todos los cambios que se han producido para que se puedan actualizar con el nuevo estado del juego.

1.8 Estudio de las tendencias actuales

El artículo *Game Logic Portability* publicado por un equipo del departamento de Ciencias de la Computación de la Universidad de Sheffield, integrado por Ahmed BinSubaih, Steve Maddock, Daniela Romano (BinSubaih, et al., 2005), centra su idea principal en separar el motor gráfico de la lógica del juego, discrepando de las tendencias actuales que fusionan estos dos subsistemas. Proponen una arquitectura donde la lógica del Juego está compuesta por las reglas y la ontología, aislando totalmente este módulo de resto del sistema. Se refieren además a un espacio de eventos que está compuesto por un adaptador, un cargador y las reglas del módulo. Donde las reglas controlan el comportamiento del juego y el adaptador se encarga de sincronizar el motor del juego con las reglas.

El cargador inicializa el Motor de reglas con plantillas para describir los atributos de los objetos reglas para gobernar comportamientos y hechos que representan objetos en el juego como son los jugadores humanos o no controlados por el jugador y puntos de caminos.

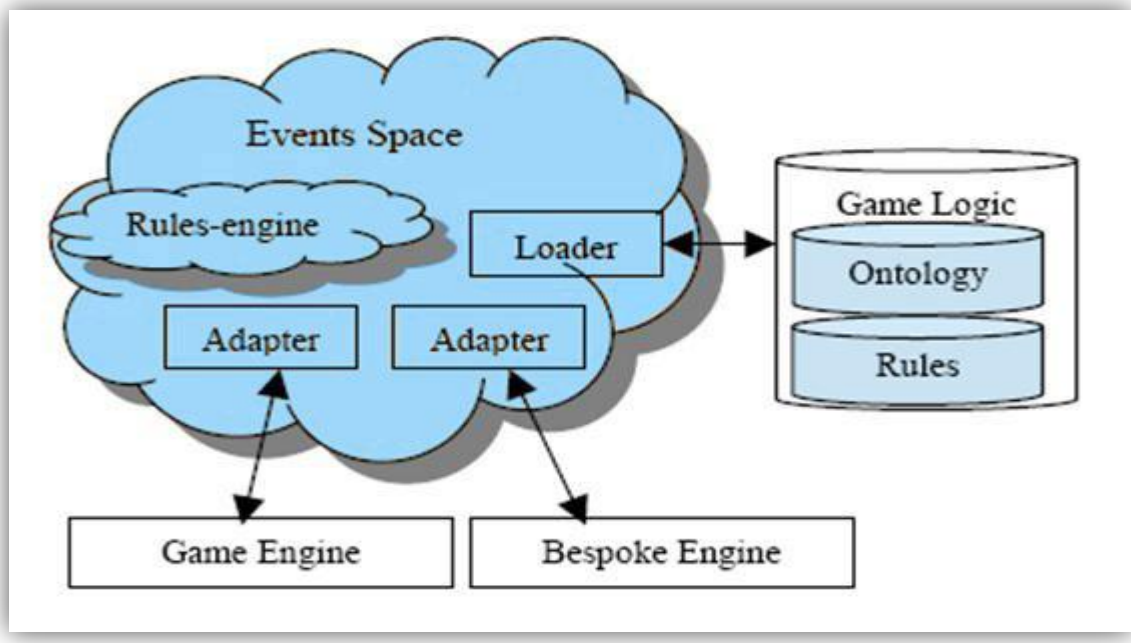

Figura 3: Panorama conceptual de la arquitectura (BinSubaih, et al., 2005)

Se analizó también el articulo *Ontologies* de los autores B. Chandrasekaran and John R. Josephson de la Universidad del estado de Ohio y V. Richard Benjamins de la universidad de Amsterdam (Chandrasekaran, 1999) que plantea que la ontología es una teoría sobre los tipos de objetos, propiedades de objetos, y las relaciones que son posibles entre los objetos de un dominio específico del conocimiento. Es la estructura del conocimiento. Así pues, el primer paso en la elaboración de un sistema eficaz es realizar un análisis ontológico efectivo del campo, o del dominio.

En el documento *A Domain-Independent Multiplayer Architecture for Training* de Ahmed BinSubaih, Steve Maddock y Daniela Romano (BinSubaih, 2004), se hace referencia a una arquitectura creada por ellos con el objetivo de tratar la flexibilidad y facilidad de la generación de hipótesis. Donde flexibilidad es la necesidad de hacer la arquitectura independiente de la investigación, cómo la lógica que es el escenario que dicta el comportamiento y está vinculado al entorno de simulación, en el que el renderizado y la creación de redes, se produce

Capítulo 2: Solución técnica

En el presente capítulo se hará una propuesta de la solución técnica para el diseño e implementación del Módulo Lógico de un videojuego partiendo del aislamiento de dicho motor de los demás subsistemas del juego, con una arquitectura compuesta fundamentalmente por un espacio de eventos, las reglas del juego y la ontología para organizar la información.

2.1- Idea General de la aplicación "Indicios Aduaneros"

La solución propuesta para satisfacer las necesidades planteadas por el cliente, es el desarrollo de un video juego en tres dimensiones donde se representaran los salones del aeropuerto internacional José Martí, las normas seguidas en el proceso de detección de indicios usado por la Aduana, que serán entonces las reglas del juego que se pretende implementar. El objetivo fundamental es detectar los pasajeros procedentes de un determinado vuelo, que incurren en casos ilícitos.

El sistema debe simular los diferentes escenarios en los que se presentan en variadas ocasiones casos delictivos y que el inspector debe detectar. Así como también se podrá evaluar el aprendizaje de dicho instructor, de forma más didáctica, a través del uso del videojuego.

El proceso de detección de indicios se divide en cuatro etapas o fases, que responde a cada uno de los salones existentes en los aeropuertos, Finger, Salón de inmigración, Pit de entrada y Salón de la Aduana. En cada uno de ellos los inspectores aduaneros verifican unas series de comportamientos y/o características físicas, psíquicas y documentales; para lo cual en determinadas circunstancias utilizan herramientas y técnicas definidas para facilitar y optimizar el trabajo.

El sistema en general deberá:

- ✓ Recrear de forma 3D los salones del aeropuerto y los pasajeros que van arribando al mismo.
- ✓ Simular el tránsito de los pasajeros a través de los 4 salones del aeropuerto.
- ✓ Permitir al jugador detallar los casos ilícitos encontrados como resultado final.

2.2- Conformación de arquitectura

Esta solución parte de la idea de separar el Módulo Lógico del resto del juego. Para tener centrado en un solo módulo todo lo referente a las reglas y la lógica del videojuego. De esa forma será reutilizable para otros proyectos y se podrá sustituir fácilmente evitando el uso de tiempo necesario para otras funciones, en el aprendizaje de un nuevo motor. Para estructurar el Módulo Lógico se utilizarán tres paquetes que distribuyan y organicen todas las funciones del subsistema: Paquete Ontológico, Espacio de Eventos, Paquete de Reglas. A continuación se explica cómo se organizan arquitectónicamente estos paquetes siguiendo un patrón de Arquitectura para una mejor organización.

"La Arquitectura del Software, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del Software" (Carnegie Mellon University, 2009).

Uno de los patrones usados para la definición de la arquitectura de software es la Arquitectura en Capas, que tiene entre sus características:

- ✓ Organización jerárquica.
- ✓ Las capas pueden ser ocultas o parcialmente opacas.
- ✓ Las capas se diseñan para que interactúen con las capas adyacentes.
- ✓ La capas se diseñan para que interactúen con las capas adyacentes, cuando esto no ocurre el estilo deja de ser puro, se pierde la flexibilidad del conjunto y complica el mantenimiento.

Para la elaboración de la arquitectura del módulo se siguió la definición del patrón de arquitectura Dos Capas.

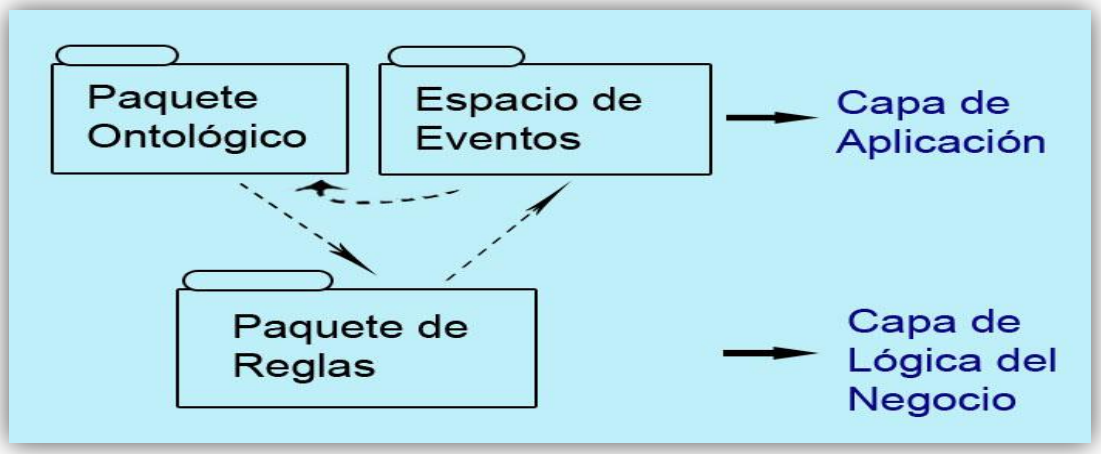

Figura 4: Diagrama de Paquetes Arquitectónicos del Módulo Lógico.

1-La primera capa que se denomina “De Aplicación” será la encargada de las funciones principales del subsistema, entre las que se encuentran: captar los eventos provenientes de los diferentes dispositivos de entrada y la implementación de las funciones del módulo. Así como la organización de la Información basado en el concepto de ontología expuesto en el capítulo anterior.

Esta capa estará compuesta por los paquetes arquitectónicos Ontológico y Espacio de Eventos. Dentro del primero se guarda la información referente a los niveles y personajes e imprentará todas las funciones referentes a estos dos conceptos. El segundo de los paquetes se menciona en el capítulo uno, pero se utilizará con una nueva organización que se ajusta más a este problema en específico. Para la presente solución dicho paquete asegura el cumplimiento de las tareas de captura de eventos provenientes del teclado y el mouse. Conformado por las estructuras CEGUI y OIS que usarán dos bibliotecas de las cuales obtienen sus nombres respectivamente. CEGUI va a captar toda la información procedente de la interfaz. Mientras que OIS capta la que proviene de los dispositivos de entrada teclado y mouse.

2-La segunda capa “Lógica del Negocio”, en la que se encuentra todo lo referente a la organización de la lógica del negocio, que corresponde a las reglas del videojuego.

Esta capa es la que contiene el Paquete de Reglas. Las reglas son las encargadas de controlar las acciones que se pueden realizar en cada nivel por un determinado personajes.

2.3 Paradigmas de programación y patrones

Después de tener divido el módulo en los paquetes arquitectónicos explicados, se organizará la programación del Motor Lógico según el libro Introducción a los videojuegos donde se explican varios paradigmas de programación, entre ellos la Programación Orientada a Objetos. En esta solución se sigue este modelo que parte de la programación con tipos de datos abstractos y le añade más funcionalidades; también se hace alusión al uso de patrones a la hora de diseñar e implementar el código de un videojuego.

2.3.1 Paradigma de programación

El paradigma de programación orientado a objeto proporciona muchos elementos para programar códigos más seguros, funcionales y reutilizables. Va más allá de todo lo que se ha hecho tradicionalmente, ayuda a tener mayor abstracción a la hora de resolver los problemas.

Tiene ventajas como:

- ✓ Aumenta la reutilización de código y de los paquetes de objetos acrecentando la productividad.
- ✓ Permiten crear jerarquías de objetos que se pueden extender más fáciles que los demás sistemas de programación.
- ✓ Para diferentes tipos de datos se pueden reutilizar las mismas líneas de código de un algoritmo. Se asegura menor probabilidad de errores por tener menos cantidad de líneas de código.
- ✓ Permite modificar un determinado algoritmo sin necesidad de reprogramar el resto de la aplicación, siempre y cuando la interfaz se mantenga.
- ✓ Facilita la programación en un equipo de trabajo. Después de concluir la fase de diseño, cada integrante del grupo puede ir programando los objetos que integran el problema con muy pocas interferencias de otro miembro.

2.3.2 Patrones de Diseño

“Un patrón de diseño consiste en la definición de un problema recurrente y de su solución simple y elegante.” (Duch i Gavaldà, et al., 2005)

Es por ello que en el caso de que un problema siga una misma línea de incógnitas a lo largo de todo su proceso de solución, se procede entonces a seguir un perfil predefinido que ayude a solucionar las dificultades que se presentan en situaciones frecuentes, sin necesidad de idear nuevamente una salida.

El uso de patrones tiene grandes ventajas entre las que se destacan:

- ✓ Proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo.
- ✓ Están basados en la recopilación del conocimiento de los expertos en desarrollo de software.
- ✓ Es una experiencia real, probada y que funciona.
- ✓ Ayuda a no cometer los mismos errores.

Se emplearon en la construcción de la arquitectura del Módulo Lógico los siguientes patrones de diseño:

En la definición de la clase controladora principal del módulo, que será la encargada de asignar responsabilidades a los otras controladoras, como el controlador de Personajes y el de Niveles. En la declaración de estas clases se aplica el **patrón controlador.** Evitando la sobrecarga de una clase pues impide que la estructura principal tenga todas las responsabilidades del módulo.

OIS es una biblioteca usada en este módulo con el objetivo de captar los eventos provenientes del teclado y el mouse y definir a que parte de la aplicación afectan directamente. Por tanto esta constantemente esperando por la entrada de un evento por parte de estos dispositivos, que es lo que define el funcionamiento del **patrón observador.**

En el diseño de la arquitectura del módulo, uno de los paquetes propuestos para integrarlo fue el de Reglas. A esta parte de la aplicación acceden en determinados momentos el paquete Ontológico y el Espacio de Eventos. Por eso se aplicaran en la implementación las características que brinda el **patrón singlenton** para crear una sola instancia de los objetos de esta clase cada vez que se solicite su acceso.

2.4 Herramientas y Lenguaje

2.4.1 Visual Studio C++

Visual C++ (también conocido como MSVC, Microsoft Visual C++) es un entorno de desarrollo integrado (IDE) para lenguajes de programación C, C++ y C++/CLI. Hace uso extensivo del *framework Microsoft Foundation Classes*, el cual es un conjunto de clases C++ para el desarrollo de aplicaciones gráficas en Windows. Cuenta con una versión Express, llamada *Microsoft Visual C++ Express Edition*, la cual es gratuita. El lenguaje de programación utilizado por esta herramienta, de igual nombre está basado en C++, y es compatible en la mayor parte de su código con este lenguaje, a la vez que su sintaxis es exactamente igual.

Los desarrolladores pueden con esta herramienta:

- ✓ 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.

2.4.2 Visual Paradigm for UML 6.4

Es una herramienta que se puede utilizar en múltiples plataformas y que es empleada en las principales IDEs. Soporta muchos lenguajes en la generación de código e Ingeniería Inversa por ejemplo Java, C++, CORBA IDL, PHP, Esquema de *Extensible Markup Language* (XML) y *Python*. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado *Unified Modeling Languaje* (UML) ayuda a una más rápida construcción de aplicaciones de calidad, brindando un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta también proporciona abundantes tutoriales, demostraciones interactivas y proyectos.

2.4.5 Lenguaje de programación C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. Es un lenguaje híbrido porque su principal objetivo es extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. Una particularidad del C++ es la posibilidad de redefinir los operadores, y de poder crear nuevos tipos que se comporten como tipos fundamentales. Permite trabajar tanto a alto como a bajo nivel. Se utilizó porque es el lenguaje de programación más óptimo para la implementación de videojuegos, y es el que se utiliza de forma estándar en todos los proyectos de simulación de la facultad al igual que el visualizador gráfico OGRE.

2.4.6 Lenguaje de modelado UML

UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

La programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

UML es un lenguaje de modelado para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

2.5 Metodología

Actualmente no existe una metodología de desarrollo de software que sea global, es decir que encierre características que puedan aplicarse a cualquier tipo de proyecto. Las características de cada proyecto conjuntamente con su equipo de desarrollo, recursos, y requisitos exigen que se escoja una que se adapte en la mayor medida posible a estas características.

Se escoge entonces la Metodología *Rational Unified Process* (RUP) que presenta las siguientes características:

- ✓ Constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.
- ✓ Dirigido por los CU, centrado en la arquitectura, iterativo e incremental.
- ✓ Cooperativo: Cliente y desarrolladores trabajan en conjunto asiduamente con una cercana comunicación.
- ✓ Sencillo: El método en sí mismo se aprende y modifica sencillamente, está bien documentado.
- ✓ Adaptable: Permite efectuar cambios de última hora.
- ✓ Es la metodología más apropiada para proyectos grandes, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas.

La metodología escogida agrupa las actividades en grupos lógicos, para ello delimita 9 flujos de trabajo. A los seis primeros se les denomina flujos de ingeniería y los tres últimos son conocidos como de apoyo. Cada uno de estos flujos pasa, en mayor o menor medida, por cuatro fases: Inicio, Elaboración y Construcción. Los Flujos de Trabajo son los siguientes, de los cuales se desarrollarán en este trabajo los cuatro primeros:

Modelo de Negocio.

Requerimientos.

Análisis y Diseño.

Implementación.

Prueba.

Despliegue.

Configuración y Administración de Cambio.

Administración de Proyecto.

Ambiente.

Capítulo 3: Modelo del negocio

En este capítulo se desarrollará el modelo del negocio que se necesita informatizar. Se proporcionará una panorámica general de cómo funciona actualmente el negocio, sirviendo esto de base para la creación del sistema, siempre partiendo de las necesidades del cliente para lograr realizar un sistema que responda a todos los requisitos que también se expondrán en el capítulo.

3.1 Reglas del negocio

- ✓ Al iniciarse la partida debe realizarse un test teórico.
- ✓ Se debe dar una descripción del vuelo a inspeccionar.
- ✓ El Inspector Aduanero (IAD) está en una posición fija.
- ✓ Puede mirar con un giro de 360 grados a la redonda.
- ✓ El IAD debe saber para cada fase del entrenamiento el tipo de indicio que debe buscar.
- ✓ El IAD solo puede darle seguimiento a 3 pasajeros a la misma vez.
- ✓ En el salón Finger no se interactúa con el Personal de Inmigración.
- ✓ En el salón de Emigración no se interactúa con el Personal de Inmigración (PI).
- ✓ Cada pasajero se debe demorar menos de 3 min en la taquilla de emigración.
- ✓ En el Pit de entrada el IA puede dar órdenes a los PI.
- ✓ En el Pit de entrada el IA puede darle órdenes a los Pasajeros.
- ✓ En el salón de espera el IA puede entrevistar a los pasajeros.
- ✓ Al finalizar el IA procesa los sospechosos.

3.2 Modelo de casos de uso del negocio

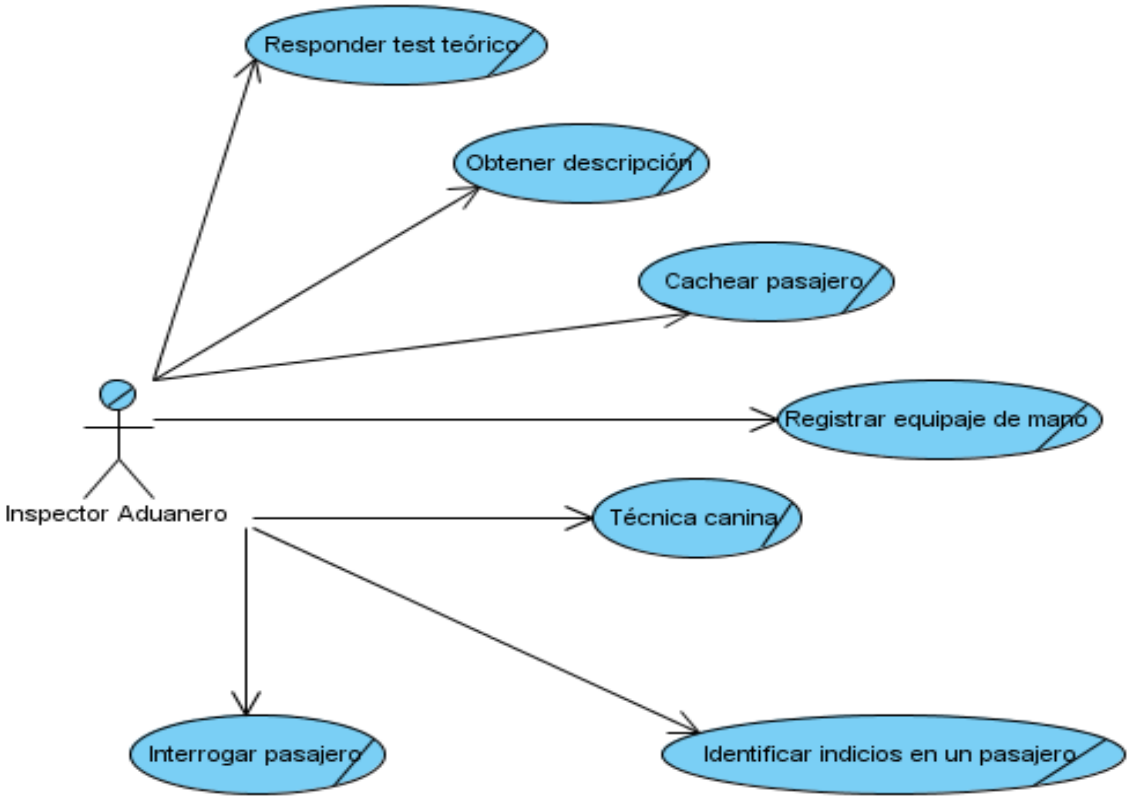

Figura 5: Diagrama de caso de uso del negocio

3.3 Descripciones de los casos de uso

Tabla 1: CUN Obtener descripción del vuelo

Caso de uso del negocio	Obtener descripción del vuelo.
Actores	IAD
Resumen	El caso de uso se inicia cuando el IAD pide la descripción del vuelo o el caso de estudio al instructor que lo evalúa. Según dicha descripción el IAD actuara.

Casos de uso asociados	-------
Flujo normal de eventos	
Acción del actor	**Respuesta del proceso de negocio**
1-El IAD pide un caso de estudio al instructor.	1.1-El instructor muestra los modos de selección para la descripción.
2-El IAD escoge modo de descripción.	2.1-Si el IAD decide modo de selección “aleatorio”, el instructor escoge aleatoriamente una descripción. 2.2-El instructor muestra la descripción escogida.
Flujo alternativo	
	2.1-Si el IAD decide modo de selección “escoger”, el instructor muestra las descripciones de los vuelos a escoger.
3-El inspector de estudio escoge una descripción.	3.1-El instructor muestra la descripción escogida.

Tabla 2: CUN Interrogar pasajero

Caso de uso del negocio	Interrogar pasajero.
Actores	IAD
Resumen	El caso de uso se inicia cuando el IAD llama al pasajero con el objetivo de realizarle una serie de preguntas y detectar algún tipo de indicio en este. Concluye cuando el pasajero se marcha o es procesado.
Casos de uso asociados	Identificar Indicios en un pasajero.

Flujo normal de eventos	
Acción del actor	**Respuesta del proceso de negocio**
1-El IAD se acerca al pasajero.	1.1-El pasajero responde al acercamiento.
2-El IAD le pide la documentación al pasajero.	2.1-El pasajero le entrega la documentación al IAD.
3-El IAD recibe los documentos.	
4-Se ejecuta el CU Identificar Indicios.	
Flujo paralelo	
4-Se le realiza una pregunta al pasajero.	4.1-El pasajero responde a la pregunta. 4.2-Durante todo este tiempo el pasajero presenta un comportamiento determinado, presenta un porte y aspecto que es analizado por el aduanero así como su comportamiento, si está nervioso, si esta intranquilo etc.
5-Si el IAD decide continuar el interrogatorio se ejecuta a partir del paso 4.	
Flujo alternativo	
5-Si el IAD decide no continuar el interrogatorio y procesar al pasajero.	
6-El IAD llama a seguridad.	
Flujo alternativo	
5-Si el IAD decide no continuar el interrogatorio y no procesar al pasajero.	
6-El IAD entrega los documentos al pasajero.	6.1-El pasajero recibe la documentación y se marcha.

Tabla 3: CUN Cachear pasajero

Caso de uso del negocio	Cachear pasajero.
Actores	IAD
Resumen	El caso de uso se inicia cuando el IAD emite la orden de registro al pasajero mediante el cacheo manual, con el objetivo de encontrar algún objeto ilícito. El caso de uso concluye cuando el Operador de Chequeo Manual libera al pasajero o se llama a seguridad.
Casos de uso asociados	-------
Flujo normal de eventos	
Acción del actor	**Respuesta del proceso de negocio**
1-El IAD emite la orden de registro.	1.2-El operador de Chequeo Manual detiene al pasajero. 1.3-El operador de Chequeo Manual procede a cachear al pasajero de forma manual. 1.4-Si el pasajero no posee ningún artículo ilícito el operador de Chequeo Manual lo libera.
Flujo alternativo	
	1.4-Si el pasajero posee algún artículo ilícito el operador de Chequeo Manual llama a seguridad.

Tabla 4: CUN Responder test teórico

Caso de uso del negocio	Responder test teórico.
Actores	IAD

Resumen	El caso de uso se inicia cuando el IAD comienza a responder las preguntas del test teórico y concluye cuando el Instructor almacena el resultado.
Casos de uso asociados	---------
Flujo normal de eventos	
Acción del actor	**Respuesta del proceso de negocio**
1-El IAD comienza a responder las preguntas del test teórico.	1.1-El Instructor calcula el resultado del test según los aciertos y desaciertos en las respuestas. 1.2-El Instructor almacena los resultados.

Tabla 5: CUN Identificar indicios

Caso de uso del negocio	Identificar indicios.
Actores	IAD
Resumen	El caso de uso se inicia cuando el IAD inspecciona detenidamente a un pasajero en busca de algún tipo de indicio. Concluye cuando este considere que no debe observarlo más.
Casos de uso asociados	---------
Flujo normal de eventos	
Acción del actor	**Respuesta del proceso de negocio**
1-El IAD inspecciona detenidamente a un pasajero.	1.1-Durante todo este tiempo el pasajero presenta un comportamiento tipo, presenta un porte y aspecto que es analizado por el aduanero así como su comportamiento, si está nervioso, si esta intranquilo etc.

2-El IAD revisa la documentación del pasajero.	
Flujo paralelo	
2-El IAD busca indicios físicos analizando el porte y aspecto del pasajero.	
Flujo paralelo	
2-El IAD busca indicios psicológicos analizando el comportamiento del pasajero.	

Tabla 6: CUN Registrar equipaje de mano

Caso de uso del negocio	Registrar equipaje de mano.
Actores	IAD
Resumen	El caso de uso se inicia cuando el IAD emite la orden de registro del equipaje de mano del pasajero, con el objetivo de encontrar algún objeto ilícito. El caso de uso concluye cuando el Técnico de Rayos X libera al pasajero o se llama a seguridad.
Casos de uso asociados	---------
Flujo normal de eventos	
Acción del actor	**Respuesta del proceso de negocio**
1-El IAD emite la orden de registro.	1.1-El Técnico de Rayos X detiene al pasajero y le pide que abra su equipaje. 1.2-El Técnico de Rayos X procede a registrar el equipaje. 1.3-Si el pasajero no posee ningún artículo ilícito el Técnico de Rayos X lo libera.

Flujo alternativo	
	1.3-Si el pasajero posee algún artículo ilícito el Técnico de Rayos X llama a seguridad.

Tabla 7: CUN Técnica canina

Caso de uso del negocio	Técnica canina.
Actores	IAD
Resumen	El caso de uso se inicia cuando el IAD emite la orden de registro del equipaje del pasajero mediante la técnica canina, con el objetivo de encontrar algún tipo de droga. El caso de uso concluye cuando técnica canina libera al pasajero o se llama a seguridad.
Casos de uso asociados	---------
Flujo normal de eventos	
Acción del actor	**Respuesta del proceso de negocio**
1-El IAD emite la orden de utilizar técnica canina.	1.1-La Técnica canina detiene al pasajero. 1.2-Se chequea el equipaje utilizando la canina. 1.3-Si al pasajero no se le detecta ningún tipo de droga se deja que este continúe.
Flujo alternativo	
	1.3-Si al pasajero se le detecta algún tipo de droga la técnica canina llama a seguridad.

3.4 Captura de requisitos

Requisitos funcionales

1. Responder test teórico.
2. Obtener descripción del vuelo.
3. Agregar pasajero a la lista de sospechosos.
4. Descartar pasajeros de la lista de sospechosos.
5. Emitir orden a un trabajador.
6. Controlar la cámara del juego.
7. Interrogar a los pasajeros.
8. Pausar/Reanudar juego.
9. Ver características.
10. Seleccionar objeto.
11. Gestionar Nivel.
12. Actualizar.
13. Iniciar simulación.

13.1. Crear nivel.

13.2. Crear utilidades.

13.3. Crear personajes.

14. Terminar simulación.

14.1. Destruir nivel.

14.2. Destruir utilidades.

14.3. Destruir personajes.

15. Observar síntomas de nerviosismo en los pasajeros.
16. Observar la forma de vestir de los pasajeros.
17. Observar alguna dificultad motora o incomodidad al caminar del pasajero.
18. Permitir observar viajeros con rasgos visibles que sugieran que es consumidor de drogas.
19. Permitir observar acciones sospechosas sobre los pasajeros.
20. Poder detectar acciones de los pasajeros para engañar al personal del Salón de Inmigración.
21. Permitir detectar acciones de los viajeros que intenten no llamar la atención.

22. Observar síntomas de nerviosismo de los pasajeros en el PIT de Entrada.
23. Observar síntomas de estrés.
24. Poder observar si los pasajeros tratan de engañar a los agentes de la aduana
25. Observar síntomas sospechosos.
26. Observar indicios sospechosos en los equipajes.
27. Observar índico sospechoso en el contenido del equipaje.
28. Permitir observar indicios sospechosos en la revisión de los documentos.
29. Permitir que se pueda reconocer a viajeros que mantengan una actitud inusual para tratar de agradar a los aduaneros y poder pasar fácilmente el salón.
30. Poder reconocer algún indicio de nerviosismo o alta sospecha durante la entrevista.
31. Permitir detectar indicios de engaño o nerviosismo a la hora de responder las preguntas de la entrevista.
32. Permitir detectar indicios sospechosos
33. Permitir detectar indicios sospechosos en la revisión de los documentos.
34. Permitir detectar acciones que indiquen que los pasajeros intentar subordinar a las fuerzas para pasar sin detectados por la entrevista.

Requisitos no funcionales

Software: Sistemas Operativos Windows XP, Ubuntu 8.10 (Hardy)

Hardware: Soporte de video para pixel shader 2.0, 1.0 GB de espacio en memoria, 1.0 GB de RAM.

Diseño e Implementación:

Herramientas*:* Visual Paradigm for UML 6.4, Visual Studio C++, Ogre3D, PhysX SDK 2.8.1.

Lenguaje de programación: C++

Bibliotecas: CEGUI, OIS, OgreMax

Usabilidad: Las personas que utilicen el juego pertenecerán al personal de la aduana y deben tener conocimientos básicos de computación.

Soporte: Debe ser compatible con Windows y Linux.

Capítulo 4: Análisis y Diseño del sistema

Después de estar terminado el flujo de trabajo de requisitos, se obtiene una vista externa del sistema. Se comienza entonces con el flujo de trabajo de análisis y diseño, donde se profundizará en el diagrama de CU correspondiente a este flujo, detallando cada uno de ellos de manera que muestren una vista interna del sistema, que sea entendible para los desarrolladores. Se determinarán las clases necesarias para realizar las funcionalidades contenidas en cada caso de uso.

4.1 Modelo de casos de uso del sistema

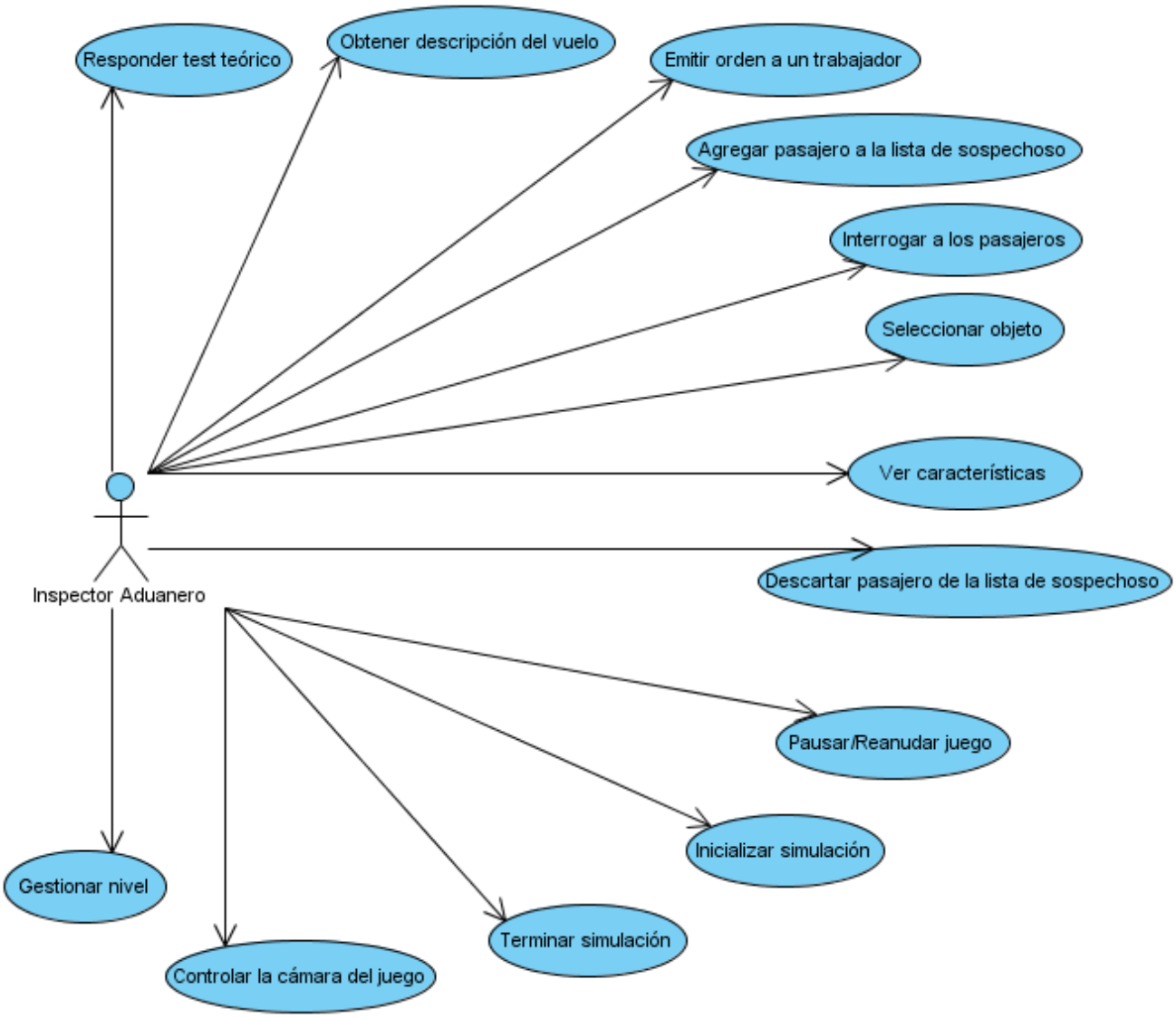

Figura 6: Diagrama de caso de uso del sistema 1

Figura 7: Diagrama de caso de uso del sistema 2

4.2 Descripciones de los casos de uso de sistema

Tabla 8: CUS Responder test teórico

Caso de uso	Responder test teórico.
Actores	IAD
Resumen	El caso de uso comienza cuando el IAD inicia una partida y presiona el botón “Mostrar test teórico” le sale una ventana con la encuesta con preguntas teóricas, y se termina cuando el inspector termina de llenar la encuesta, que presione un botón para salir.
Casos de usos asociados	**-------**

Flujo normal de eventos	
Acción del Actor	**Respuesta del sistema**
1-El IAD presiona el botón “Mostrar test teórico”.	1.1-El sistema muestra en pantalla una encuesta pre-elaborada.
2-El IAD responde las preguntas de la encuesta.	2.1-El sistema calcula y procesa el resultado del cuestionario.

Tabla 9: CUS Obtener descripción del vuelo

Caso de uso	Obtener descripción del vuelo.
Actores	IAD
Resumen	El caso de uso comienza cuando el IAD presiona el botón para obtener la descripción del vuelo, y termina cuando el inspector cierra esta ventana.
Casos de usos asociados	**--------**

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-El IAD presiona el botón para obtener la descripción del vuelo.	1.1-El sistema muestra la descripción del vuelo. 1.2-El sistema cierra la ventana de descripción del juego y comienza la partida.

Tabla 10: CUS Emitir orden a un trabajador

Caso de uso	Emitir orden a un trabajador.
Actores	IAD
Resumen	El caso de uso comienza cuando un IAD le da una orden a un trabajador y culmina con la ejecución de la misma.
Precondiciones	Se ha seleccionado al menos un objeto.
Casos de usos asociados	**--------**
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-El IAD presiona el botón Emitir Orden.	1.1-El sistema busca con el id dado por el CU Seleccionar Objeto, cual es el trabajador al que se le dará la orden. 1.2-Si encuentra al trabajador el sistema muestra una ventana con las posibles órdenes a emitir.
2-El IAD selecciona la orden que desea emitir al trabajador.	2.1-El sistema asigna la orden al trabajador.

Flujo alterno de eventos	
	1.2-Si no lo encuentra termina el CU.

Tabla 11: CUS Agregar pasajero a la lista de sospechosos

Caso de uso	Agregar pasajero a la lista de sospechosos.
Actores	IAD
Resumen	El caso de uso comienza cuando el IAD presiona el botón de agregar sospechoso y termina cuando se añade a la lista y se actualiza este componente.
Precondiciones	Se ha seleccionado al menos un objeto.
Casos de usos asociados	**--------**
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-El IAD presiona el botón de agregar sospechoso.	1.1-El sistema buscar en la lista de sospechosos al pasajero seleccionado para ver si no está. 1.2- Si no los encuentra el sistema agrega el sospechoso. 1.3-El sistema muestra una ventana con la lista de sospechosos actualizada.
Flujo alterno de eventos	
	1.2- Si lo encuentra en la lista se termina el CU.

Tabla 12: CUS Descartar pasajero de la lista de sospechosos

Caso de uso	Descartar pasajero de la lista de sospechosos.
Actores	IAD
Resumen	El caso de uso comienza cuando el IAD presiona el botón para descartar un pasajero de la lista de sospechosos, y termina cuando ha eliminado el personaje de la lista de sospechosos y la actualización de esta componente.
Casos de usos asociados	**--------**

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-El IAD presiona el botón para descartar un pasajero de la lista de sospechosos.	1.1-El sistema muestra una ventana con la lista actual de sospechosos.
2-El IAD elimina el o los sospechosos de la lista.	2.1-El sistema elimina el sospechoso seleccionado. 2.2-El sistema muestra una ventana con la lista de sospechosos actualizada.

Tabla 13: CUS Interrogar a los pasajeros

Caso de uso	Interrogar a los pasajeros.
Actores	IAD
Resumen	El caso de uso comienza cuando un IAD presiona el botón para interrogar pasajero, el sistema muestra una ventana con las posibles preguntas a realizar y termina con la realización de la entrevista.
Precondiciones	Se ha seleccionado al menos un objeto.

Casos de usos asociados	--------
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-En IAD presiona el botón para interrogar pasajero seleccionado.	1.1-El sistema muestra una ventana con las posibles preguntas a realizar.
2-Selecciona la pregunta que desea hacer.	2.1-El sistema le asigna la pregunta al pasajero, y si el IAD desea seguir preguntando vuelve al paso 1.1 del CU. 2.2-El sistema asigna la última pregunta realizada por el IAD.

Tabla 14: CUS Pausar/Reanudar juego

Caso de uso	Pausar/Reanudar juego.
Actores	IAD
Resumen	El caso de uso comienza cuando el IAD le da Pausar o Reanudar al juego, y termina cuando el sistema inicia o pausa cada uno de los componentes.
Casos de usos asociados	--------
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-El IAD presiona el botón Pausar o Reanudar Juego.	1.1-En caso de que se presione Pausar Juego el sistema ejecuta la sección 1, y la 2 en caso de que

	sea Reanudar Juego.
Sección 1 Pausar Juego	
	1.1-El sistema salva todos los datos que se estén ejecutando o modificando en ese momento y detiene todas las funcionalidades del juego. 1.2-El sistema pausa la partida. 1.3-El sistema deja de actualizar.
Sección 2 Reanudar Juego	
	2.1-El sistema carga todos los datos que se habían salvado. 2.2-El sistema reanuda la partida. 2.3-El sistema vuelve a actualizar la partida.

Tabla 15: CUS Controlar cámara del juego

Caso de uso	Controlar cámara del juego.
Actores	IAD
Resumen	El caso de uso comienza cuando un IAD decide modificar la posición actual de la cámara, se remite al botón que brinda esta opción, ubica la nueva posición. El caso de uso termina con el nuevo posicionamiento de la cámara.
Casos de usos asociados	**--------**
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-El IAD decide modificar la posición actual	1.1-El sistema modifica la cámara del juego según

de la cámara y presiona el botón que brinda dicha opción.	el movimiento realizado por el jugador. 1.2-El sistema posiciona nuevamente la cámara del juego.

Tabla 16: CUS Iniciar simulación

Caso de uso	Iniciar simulación.
Actores	IAD
Resumen	El caso de uso comienza cuando el IAD le da clic en el botón “Iniciar partida”, que se crean las utilidades, el nivel y los pasajeros y termina cuando ya estos han sido creados.
Casos de usos asociados	**---------**
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-El IAD presiona el botón de iniciar partida.	1.1-El sistema inicializa las variables y estados globales del juego, se generan los personajes, se crean las utilidades y se inicializa el nivel. 1.2-Se inicializa la partida completa y esta lista para comenzar a jugar.

Tabla 17: CUS Terminar simulación

Caso de uso	Terminar simulación.
Actores	IAD
Resumen	El caso de uso comienza cuando el IAD presiona el botón de terminar partida, que se empieza a destruir el nivel, las utilidades y los personajes

	y termina cuando se hayan destruido.
Casos de usos asociados	**----------**
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-El IAD presiona el botón de terminar partida.	1.1-El sistema destruye las variables y estados globales del juego, los personajes, las utilidades y el nivel. 1.2-Se destruyen las utilidades, se muestra la terminación de la partida y se muestra nuevamente el menú.

Tabla 18: CUS Seleccionar objeto

Caso de uso	Seleccionar objeto.
Actores	IAD.
Resumen	El caso de uso comienza cuando un IAD le da clip a cualquiera de los personajes y termina cuando se obtiene el nombre del objeto seleccionado.
Precondiciones	**----------**
Casos de usos asociados	**----------**
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-El IAD le da clip a cualquiera de los	1.1-El sistema busca el nombre del objeto

personajes.	seleccionado. 1.2-El sistema busca, con el objeto encontrado el id del mismo. 1.3-El sistema guarda el id del objeto seleccionado.

Tabla 19: CUS Ver características

Caso de uso	Ver características.
Actores	IAD
Resumen	El caso de uso comienza cuando un IAD presiona el botón de observar características de un pasajero seleccionado, y termina cuando se hayan mostrado dichas características.
Precondiciones	Se ha seleccionado al menos un objeto.
Casos de usos asociados	---
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-El IAD presiona el botón de observar características de un pasajero seleccionado.	1.1-El sistema muestra la opción para escoger que tipo de características desea ver.
2-El IAD selecciona la característica deseada.	2.1-El sistema busca el personaje seleccionado en la lista. 2.2-El sistema obtiene las características del personaje encontrado. 2.3-Se muestran las características del pasajero.

Tabla 20: CUS Gestionar niveles

Caso de uso	Gestionar niveles.
Actor	IAD
Resumen	El caso de uso comienza el IAD solicita el cambio de nivel y termina cuando se hayan desactivados las opciones que no están permitidas para el nuevo nivel.
Precondiciones	Debe haberse inicializado un x nivel.
Casos de usos asociados	**--------**
Restricción especial del caso de uso	El sistema puede cambiar automáticamente de nivel si se termina el tiempo dado y el usuario no ha terminado en el nivel que esta. Esta sería otra forma de iniciar el Caso de Uso.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1-El caso de uso comienza el IAD solicita el cambio de nivel.	1.1-El sistema carga el próximo nivel con todas las opciones existentes. 1.2-El controlador de reglas le dice al sistema cuales son las acciones que están permitidas para ese nivel. 1.3-El sistema desactiva los botones que dan acceso a las acciones que se pueden realizar en este nivel.

Tabla 21: CUS Actualizar

Caso de uso	Actualizar.	
Actor	Aplicación	
Resumen	El caso de uso comienza cuando se inicializa la simulación y termina cuando de termina la partida.	
Precondiciones	**-------**	
Casos de usos asociados	**--------**	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
1-El juego pide al módulo la actualización del frame que necesita que se actualice en ese momento.		1.1-El sistema actualiza todos los componentes correspondientes al nivel y los personajes. 1.2-El sistema continúa mostrando el juego con la nueva actualización.

4.3 Diagramas de clases del análisis

Figura 8: DCA "Responder test teórico"

Figura 9: DCA "Obtener descripción del vuelo"

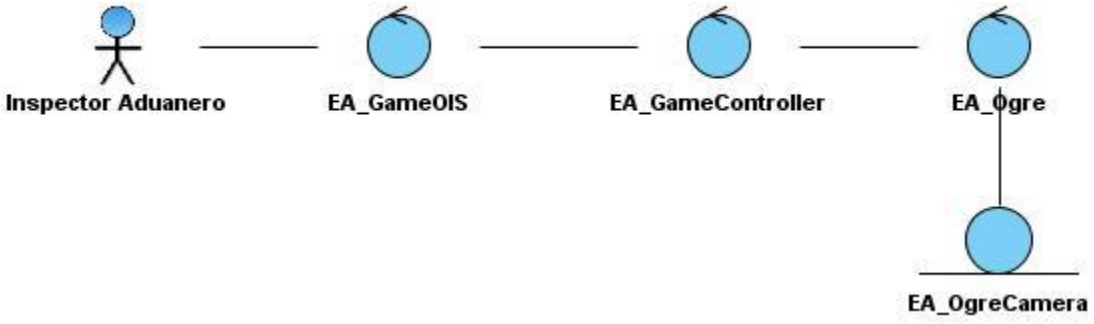

Figura 10: DCA "Controlar cámara de juego"

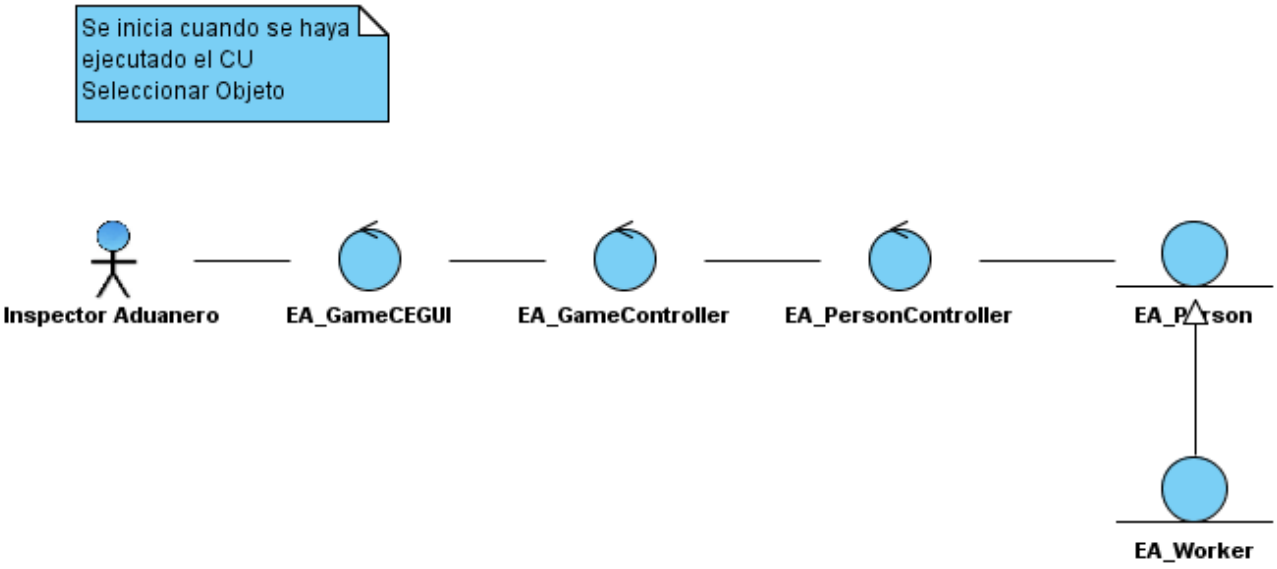

Figura 11: DCA "Emitir orden a un trabajador"

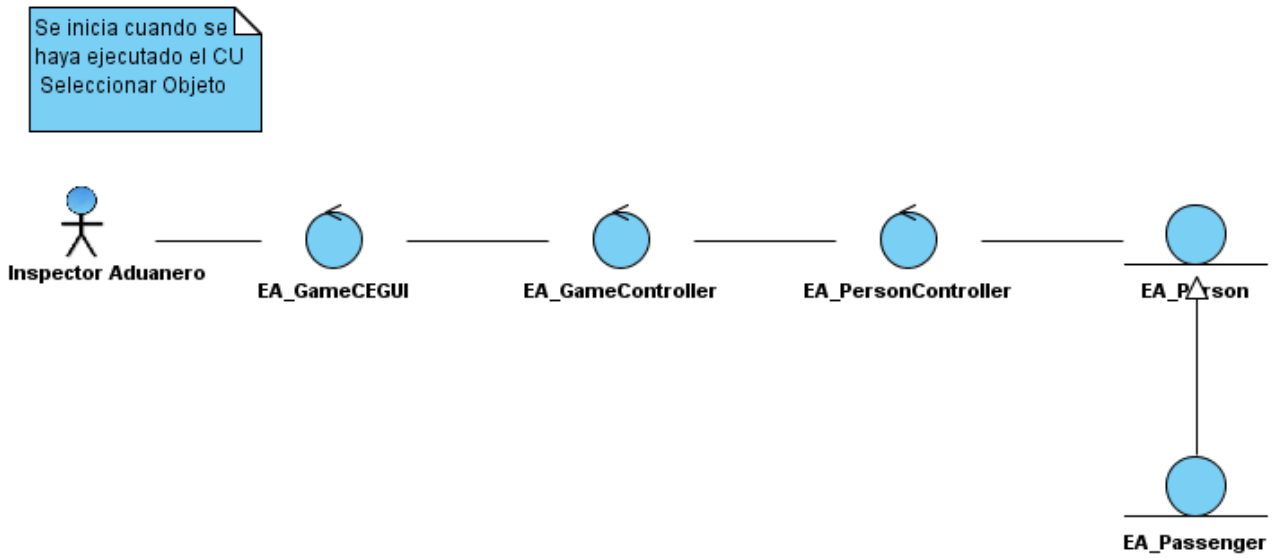

Figura 12: DCA "Agregar sospechoso a la lista de pasajeros"

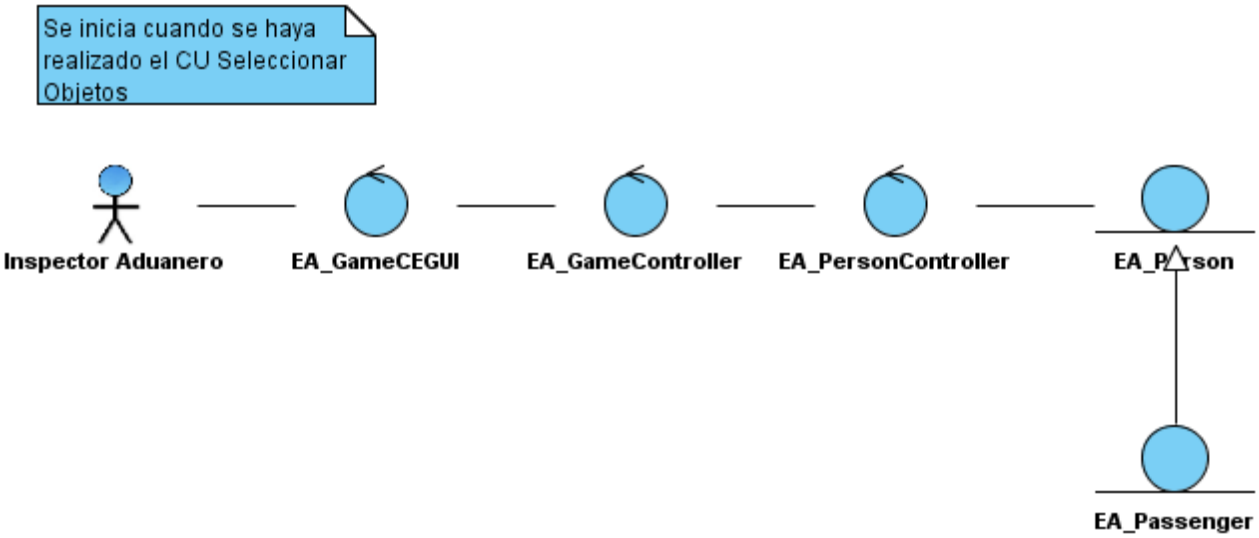

Figura 13: DCA "Interrogar pasajero"

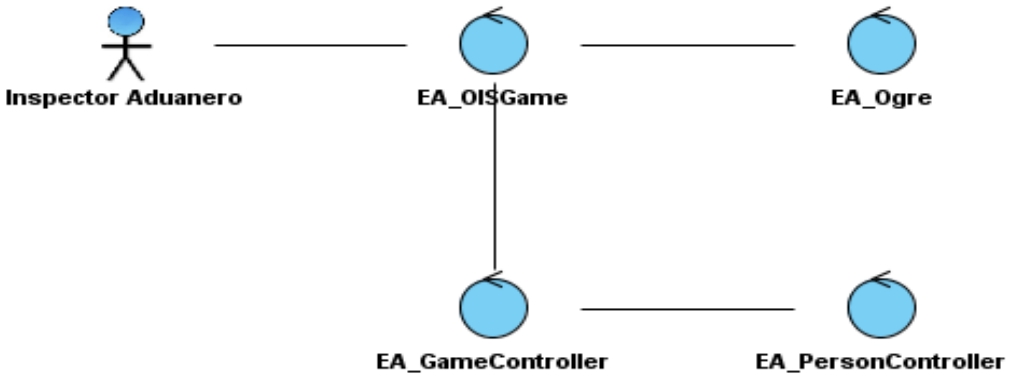

Figura 14: DCA "Seleccionar Objeto"

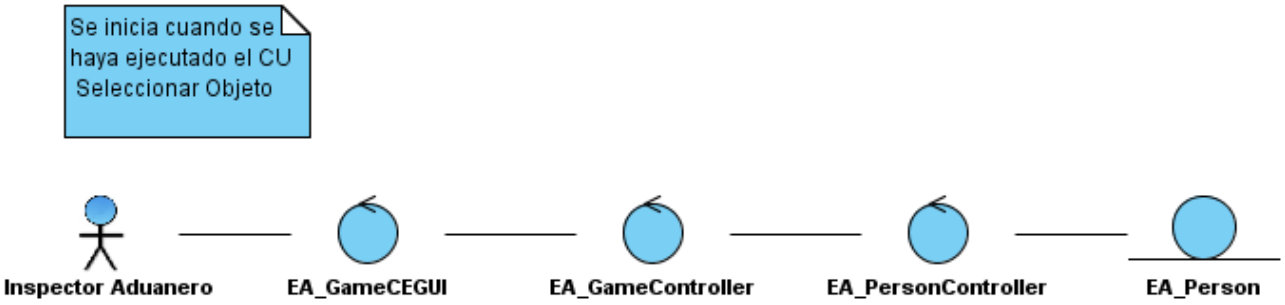

Figura 15: DCA "Ver características"

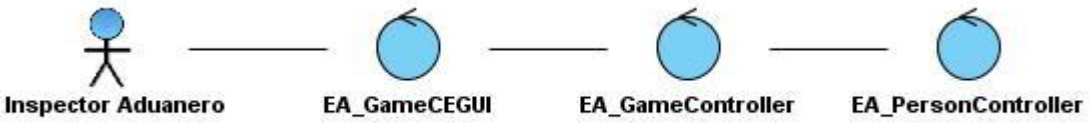

Figura 16: DCA "Descartar sospechoso de la lista"

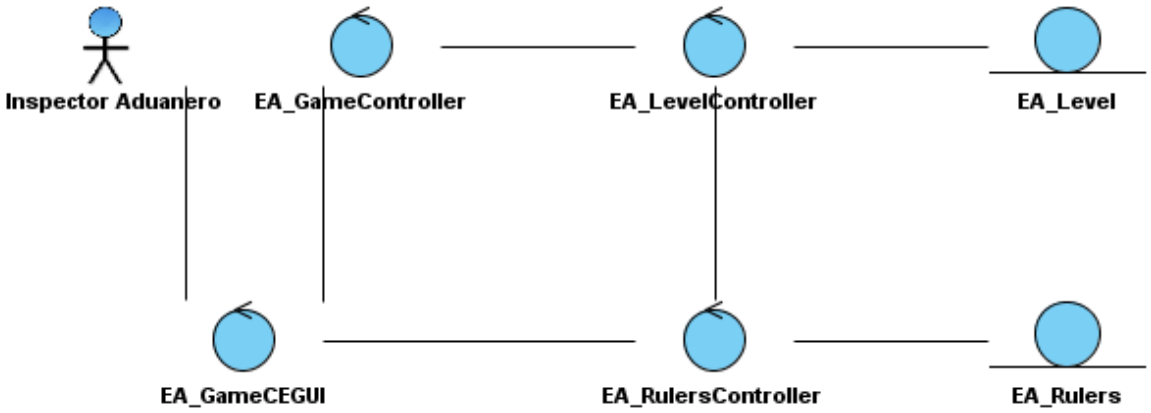

Figura 17: DCA "Gestionar Nivel"

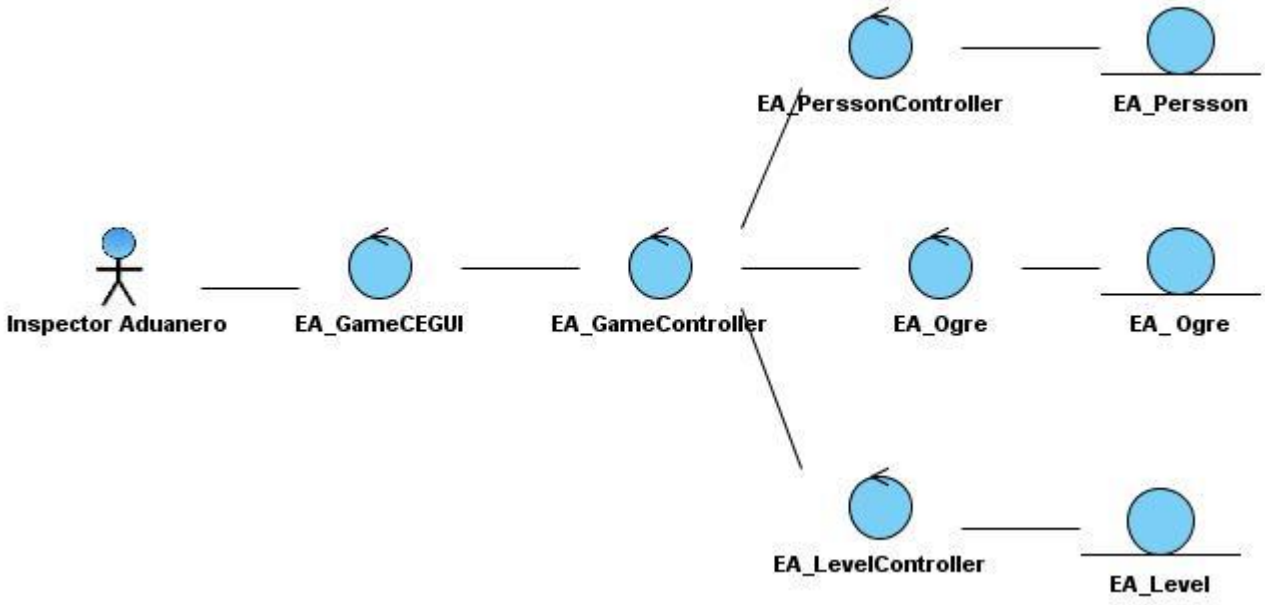

Figura 18: DCA "Pausar/Reanuda juego"

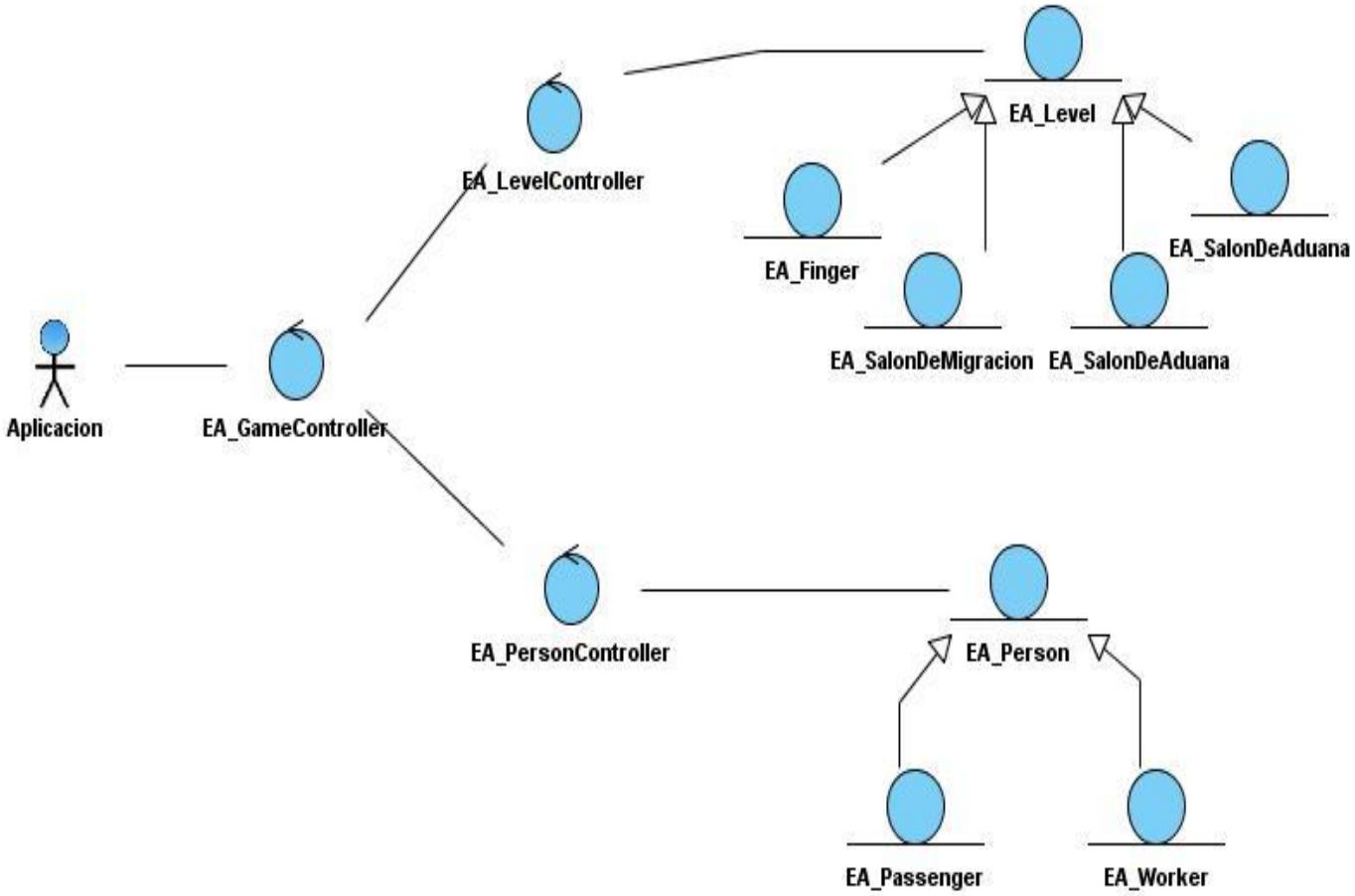

Figura 19: DCA "Actualizar"

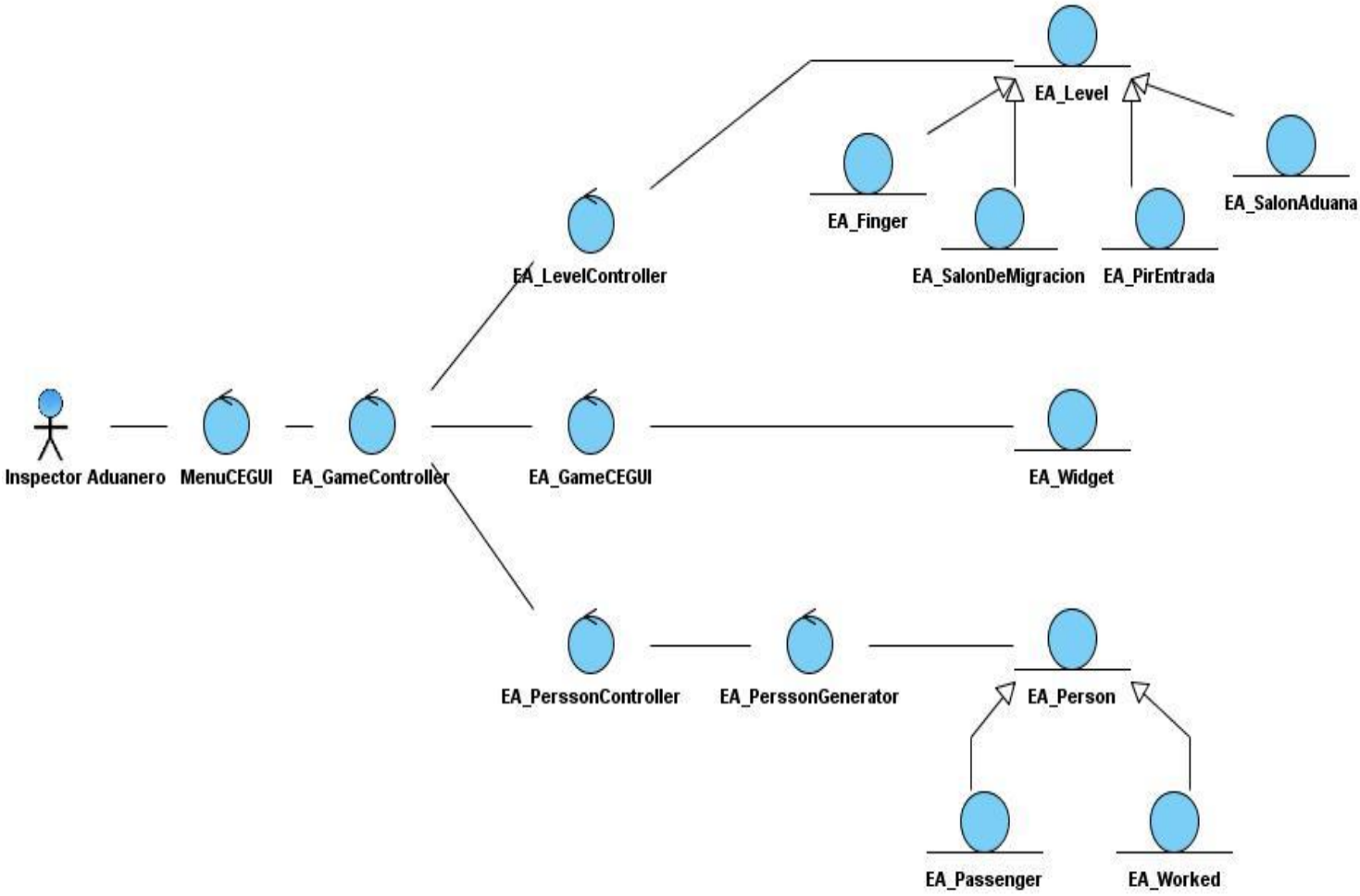

Figura 20: DCA "Iniciar simulación"

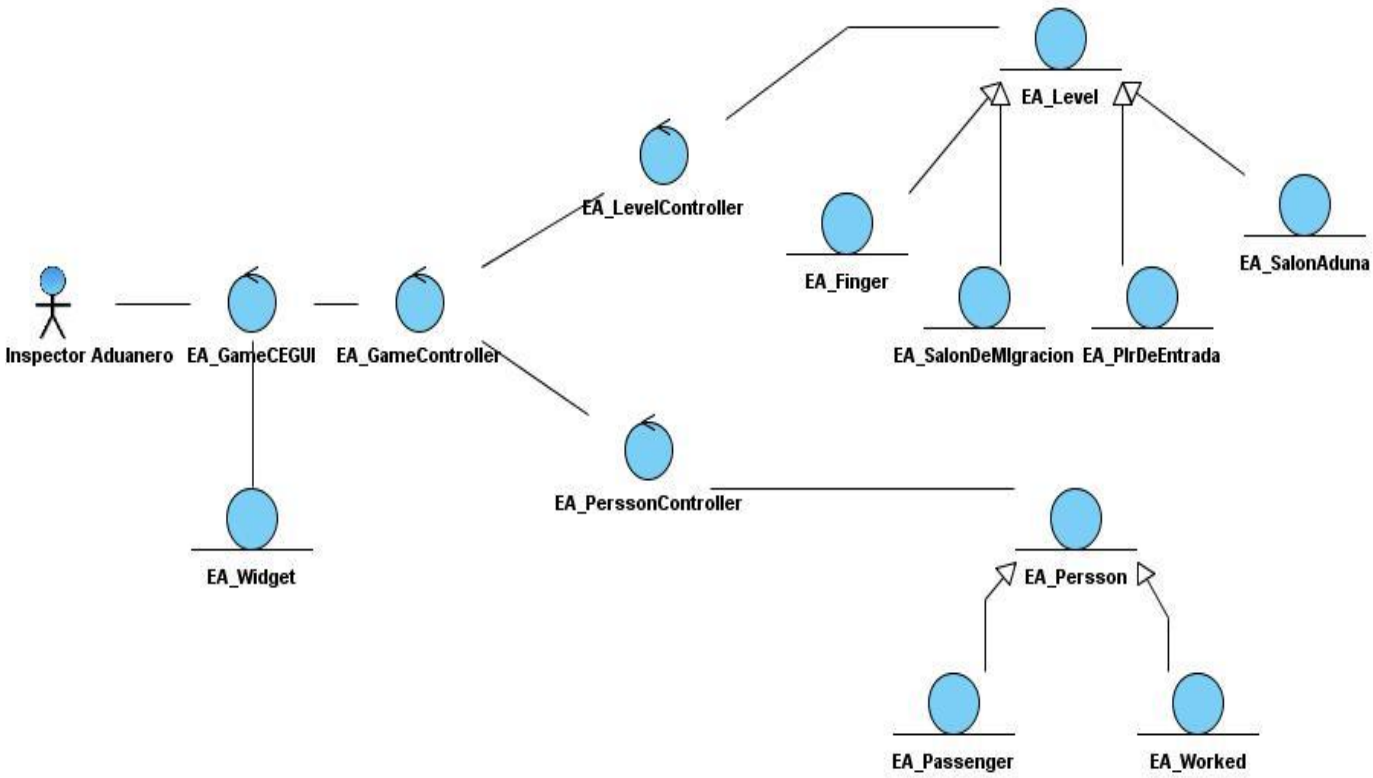

Figura 21: DCA "Terminar simulación"

4.4 Diagramas de clases del diseño

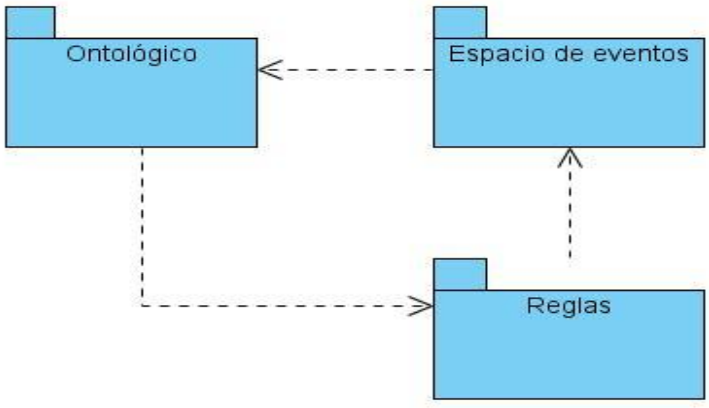

Figura 22: Diagrama de Paquete de Clases del Diseño

4.4.1 Paquete "Ontológico"

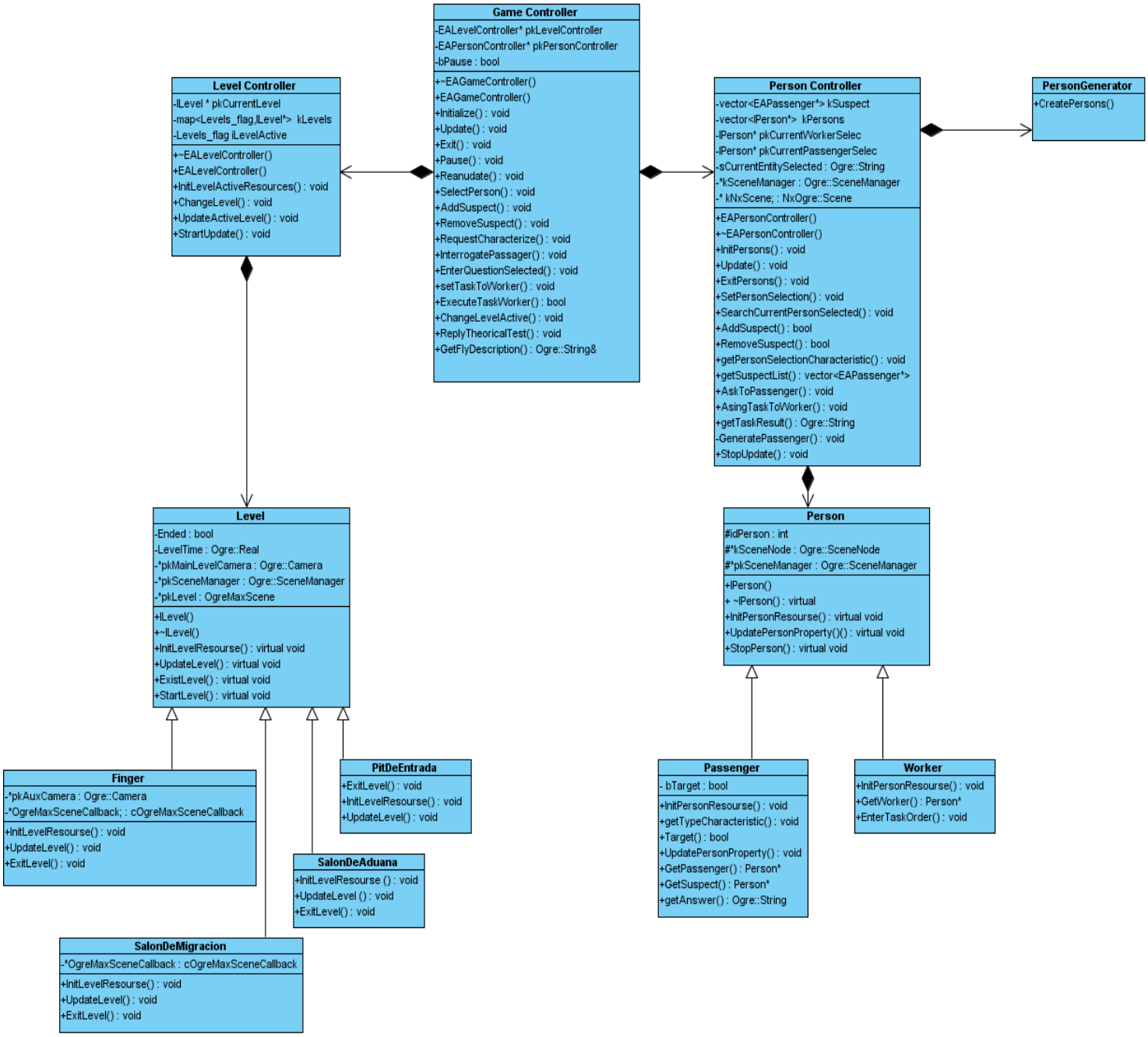

Figura 23: Diagrama de Clases de Paquete "Ontológico"

4.4.2 Paquete "Espacio de eventos"

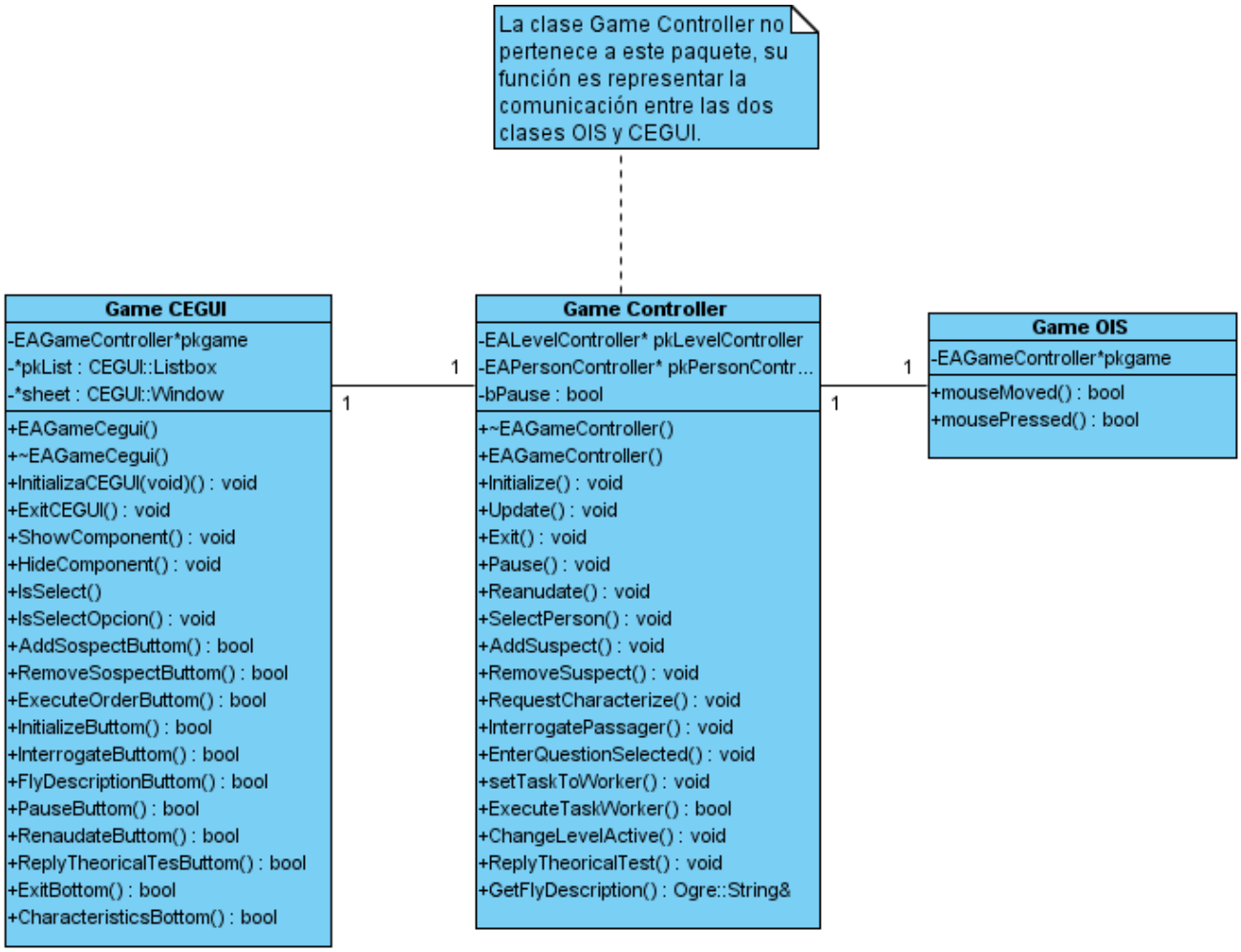

Figura 24: Diagrama de Clases de Paquete "Espacio de Eventos"

4.4.3 Paquete “Reglas”

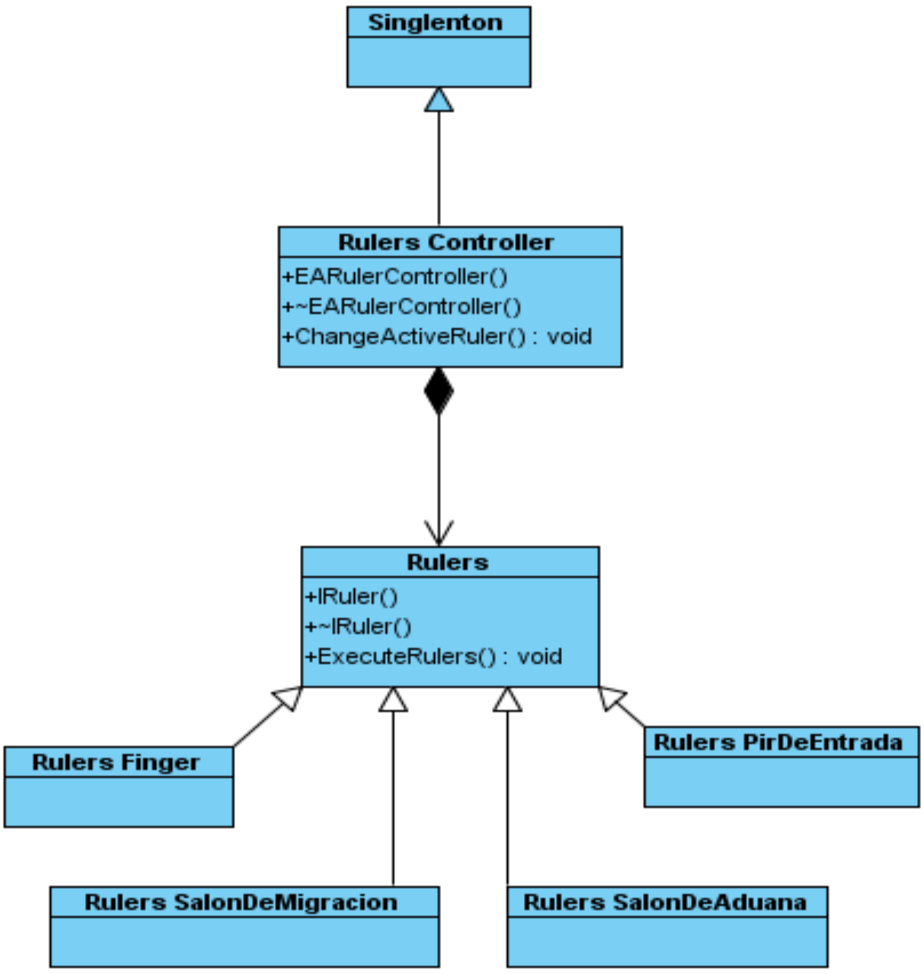

Figura 25: Diagrama de Clases de Paquete “Reglas”

4.5 Diagramas de iteración del diseño

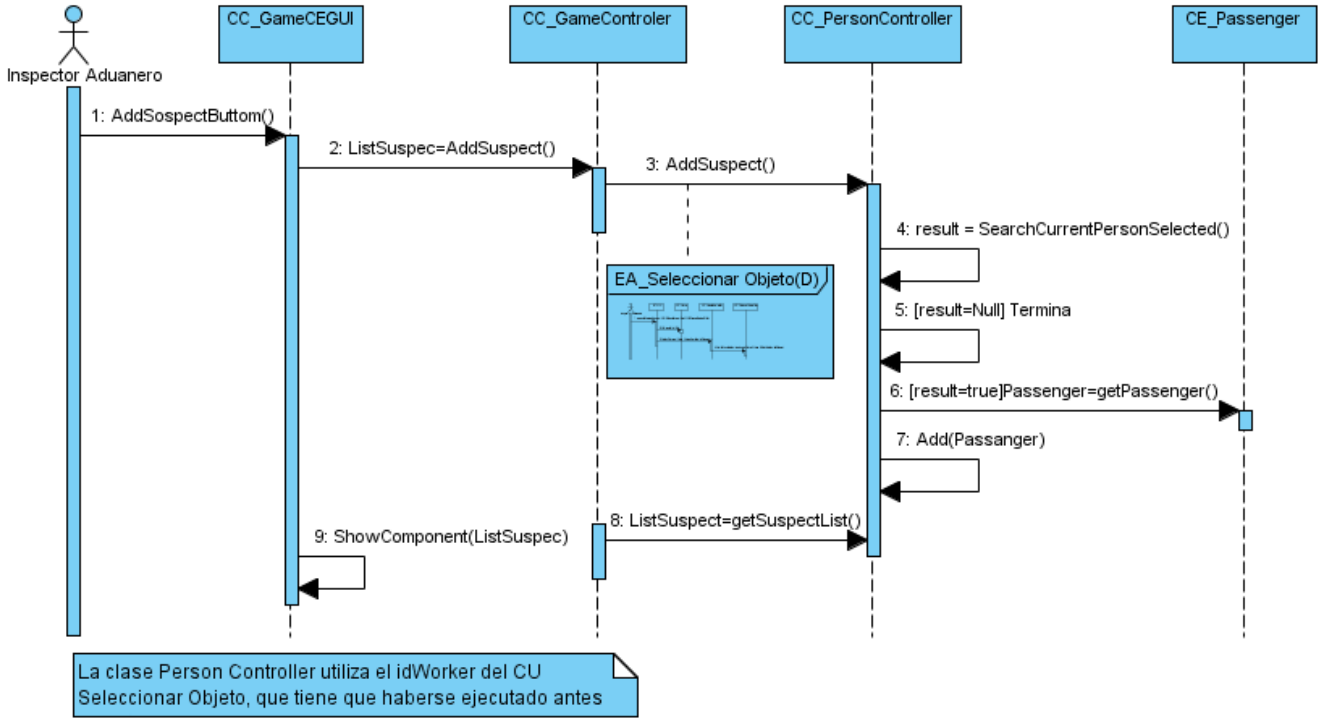

Figura 26: DCD "Agregar sospechoso a la lista"

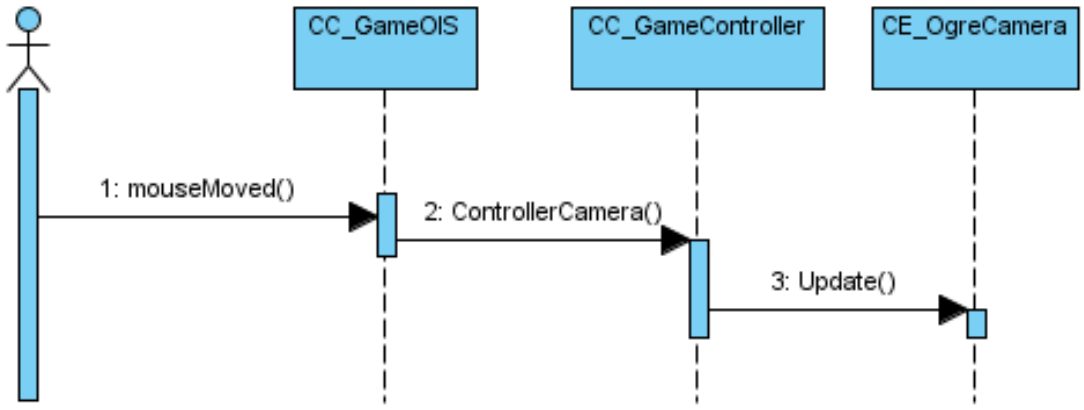

Figura 27: DCD "Controlar cámara del juego"

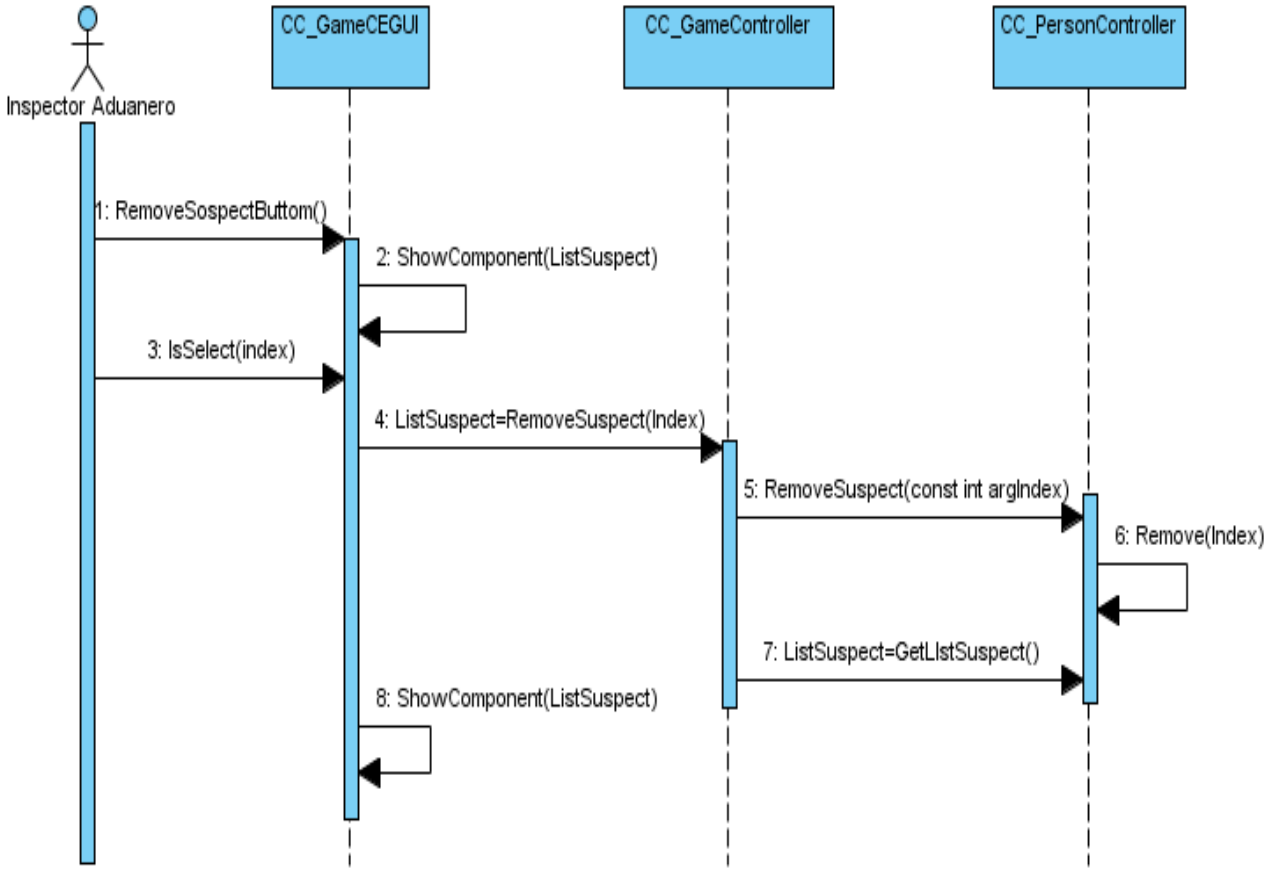

Figura 28: DCD "Descartar sospechoso"

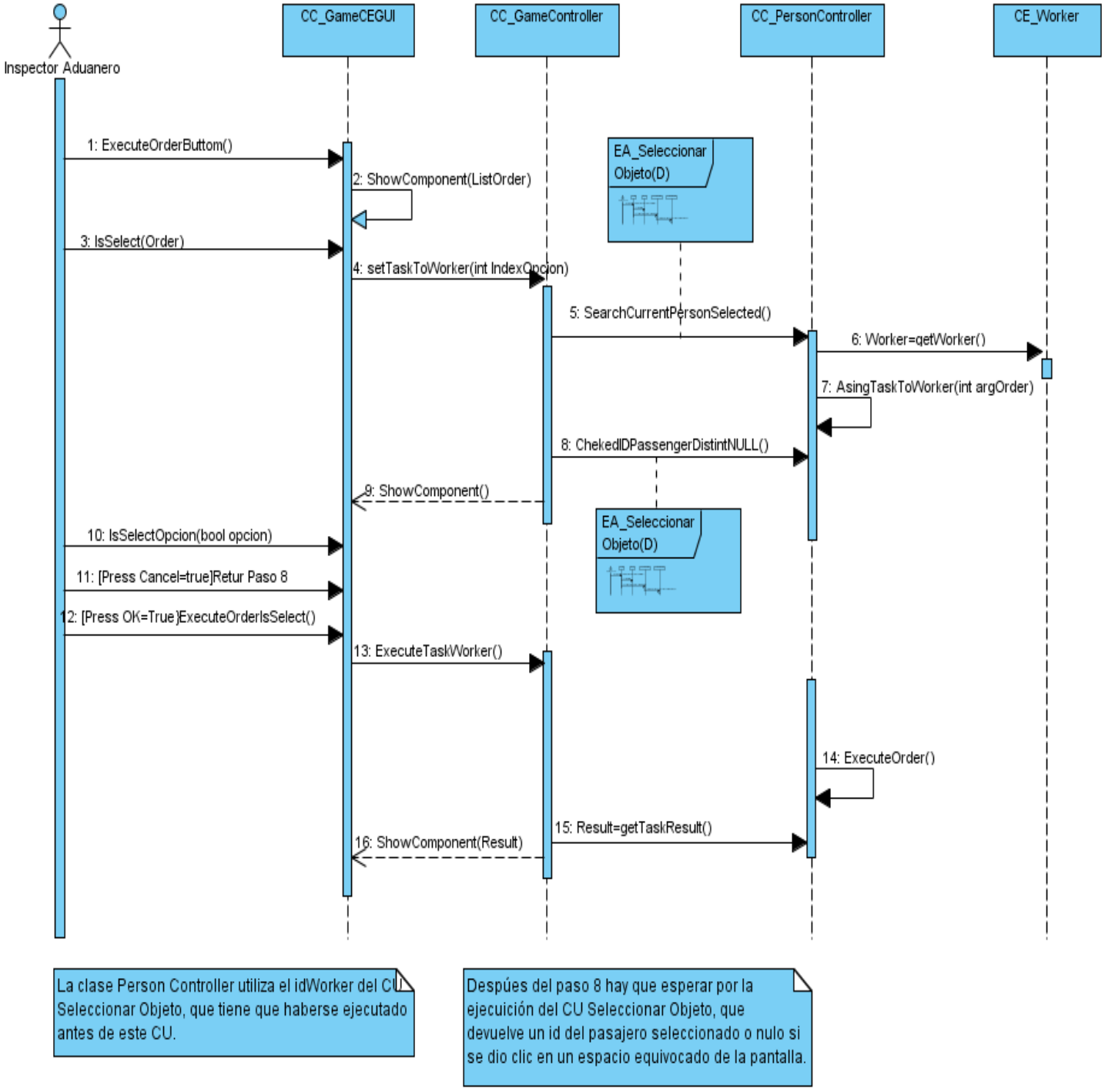

Figura 29: DCD "Emitir orden a un trabajador"

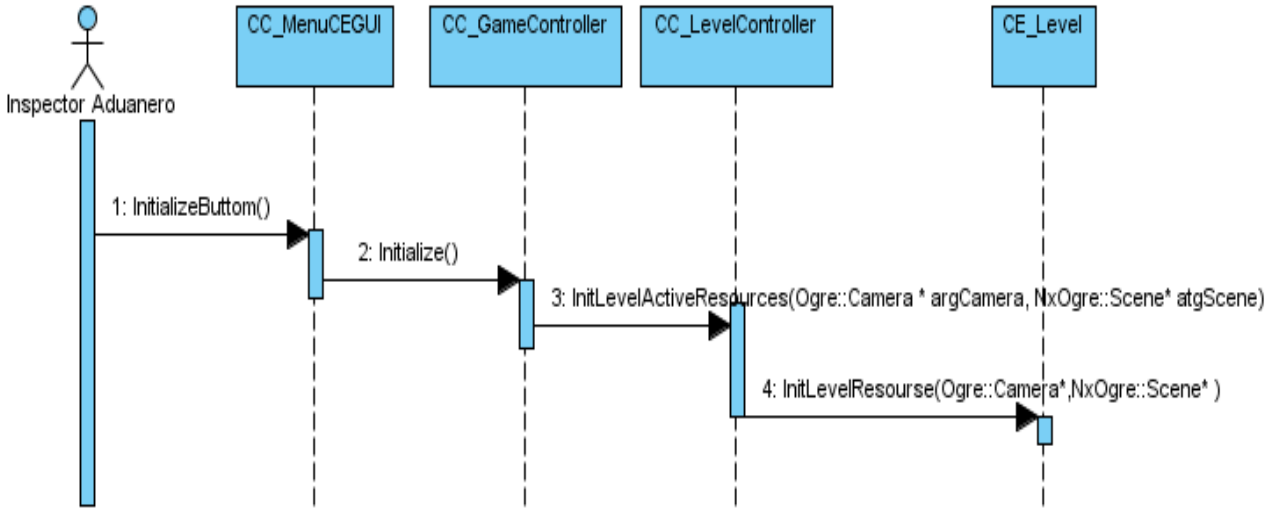

Figura 30: DCD "Iniciar simulación (1)"

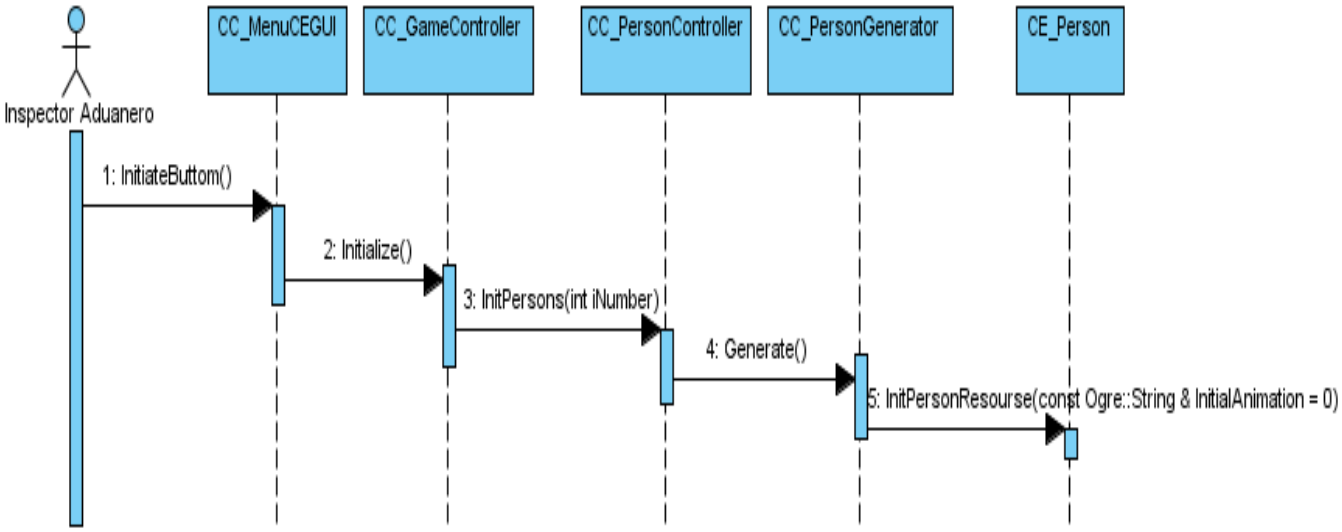

Figura 31: DCD "Iniciar simulación (2)"

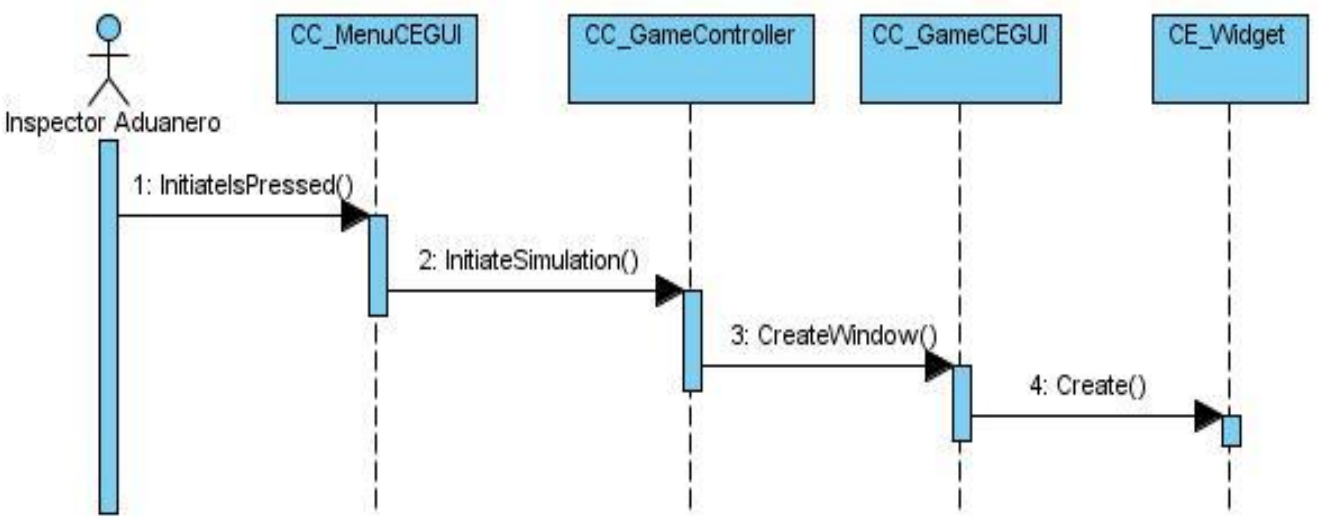

Figura 32: DCD "Iniciar simulación (3)"

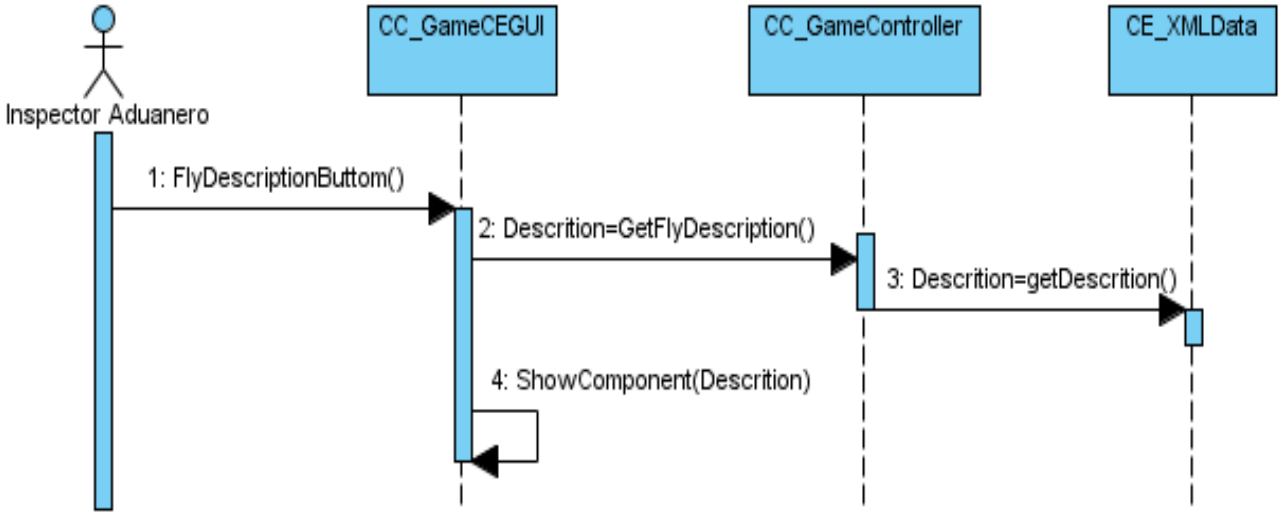

Figura 33: DCD "Obtener descripción del vuelo"

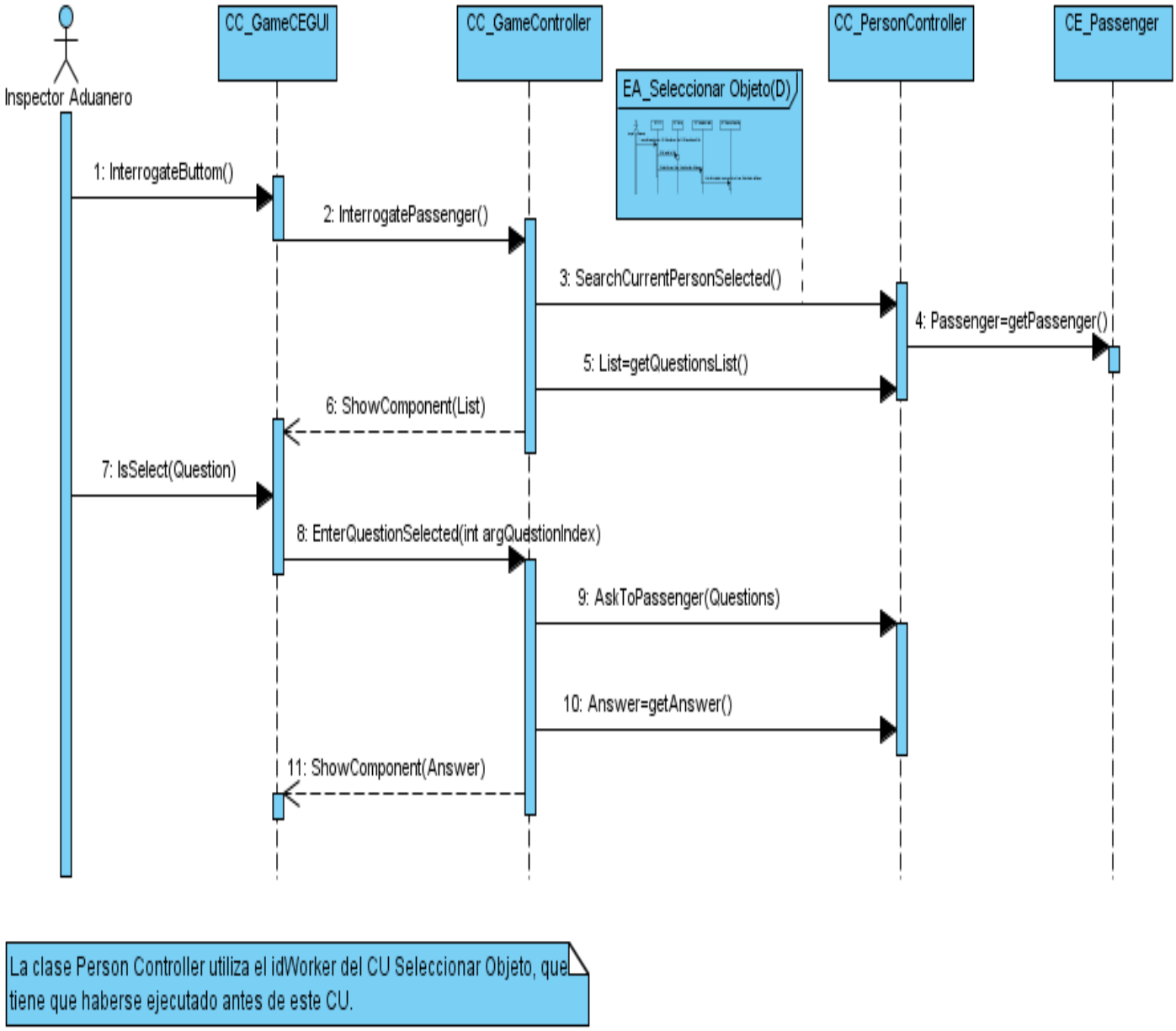

Figura 34: DCD "Interrogar un pasajero"

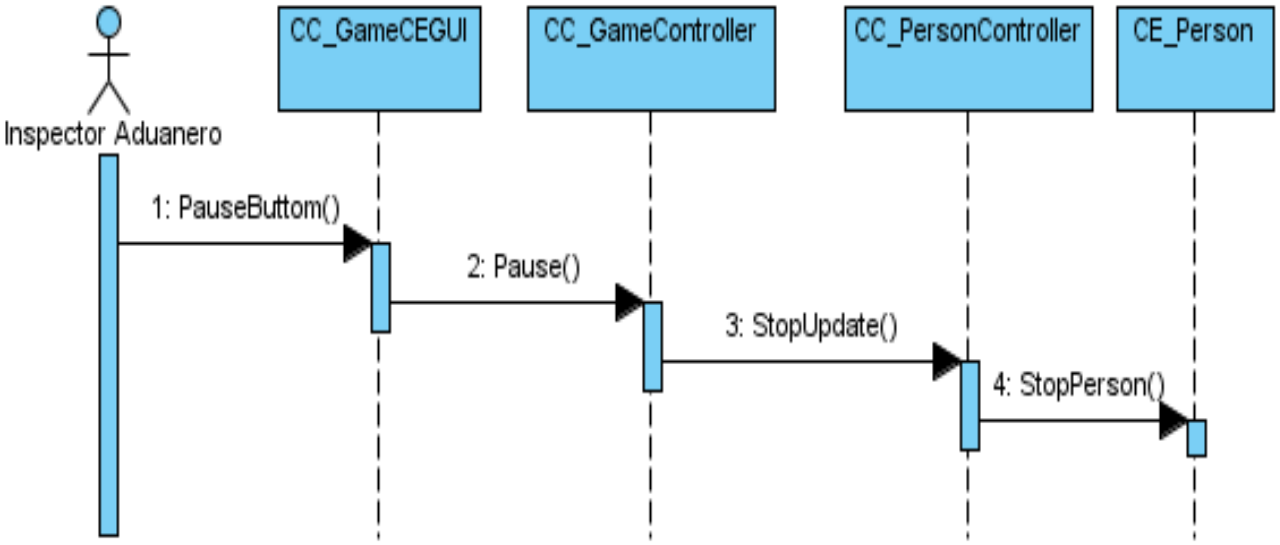

Figura 35: DCD "Pausar (1)"

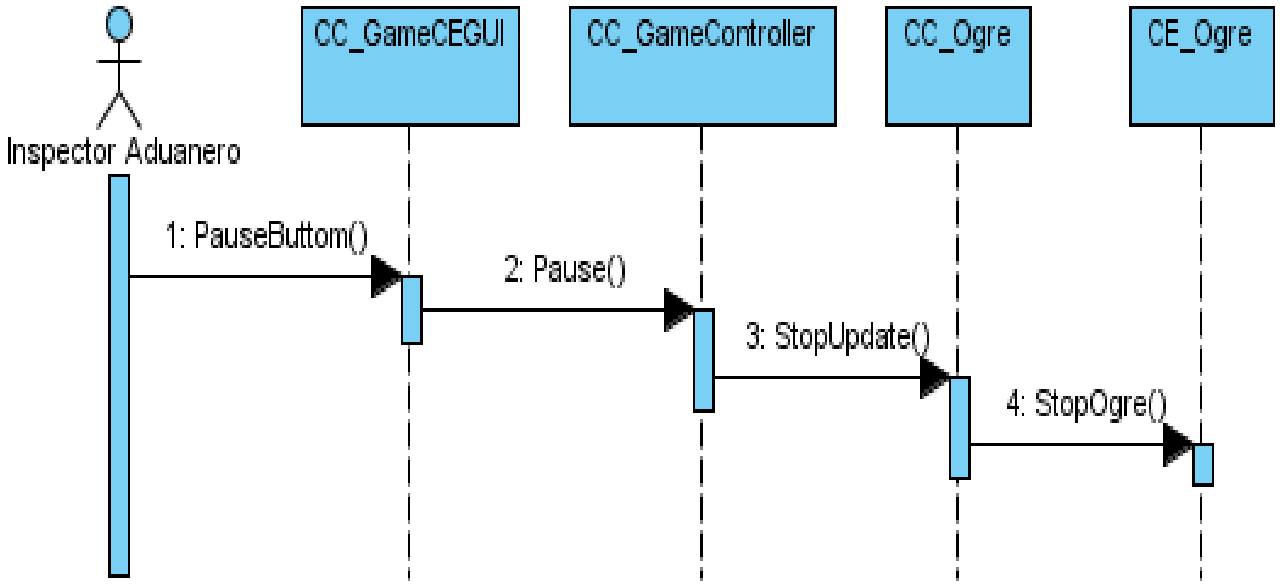

Figura 36: DCD "Pausar (2)"

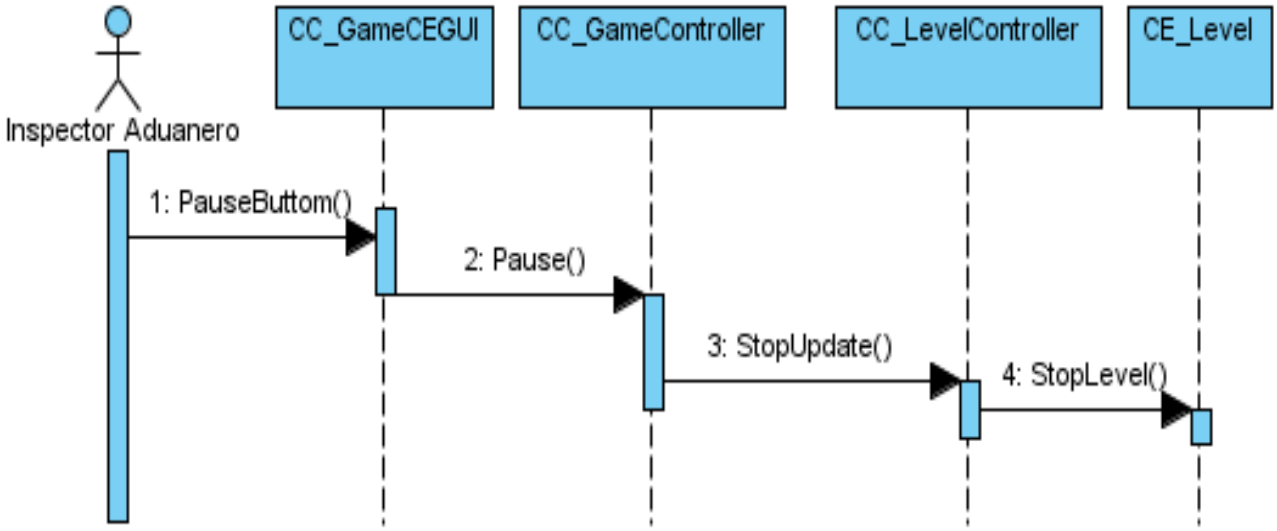

Figura 37: DCD "Pausar (3)"

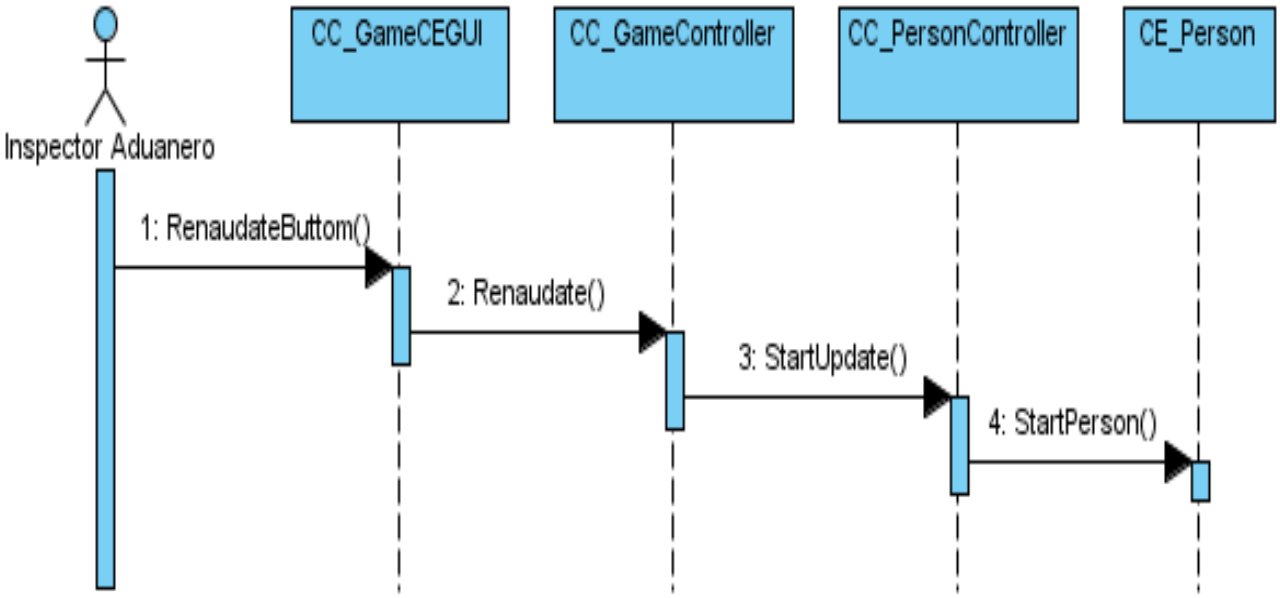

Figura 38: DCD "Reanudar (1)"

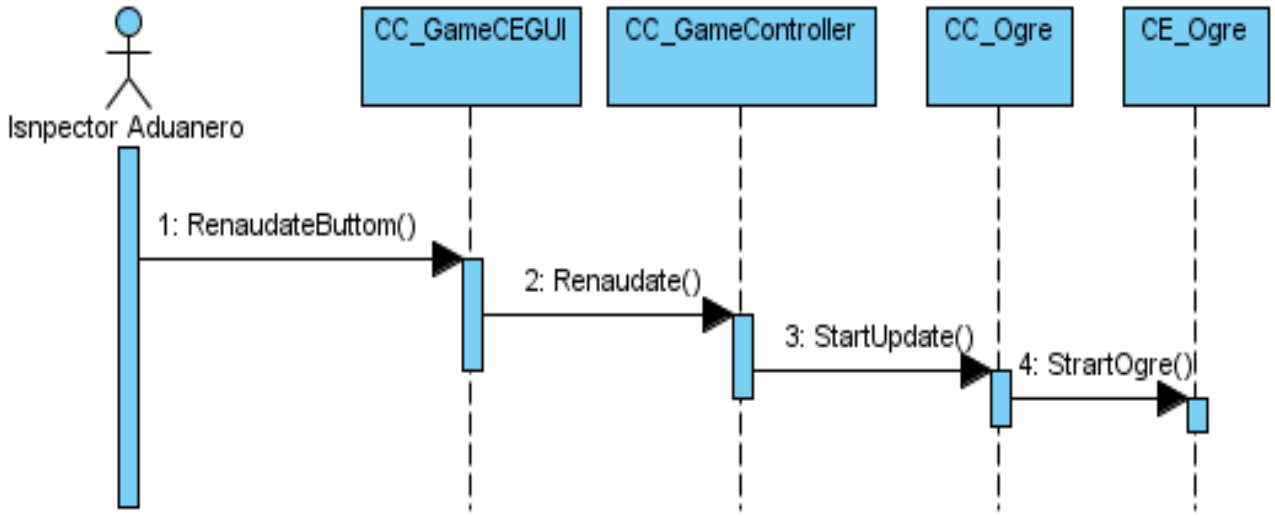

Figura 39: DCD "Reanudar (2)"

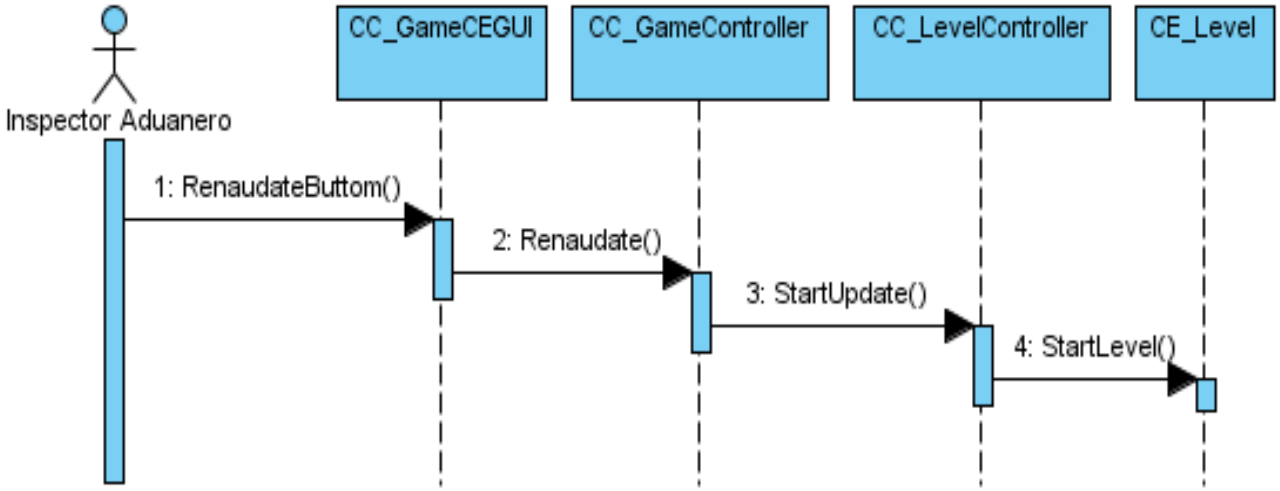

Figura 40: DCD "Reanudar (3)"

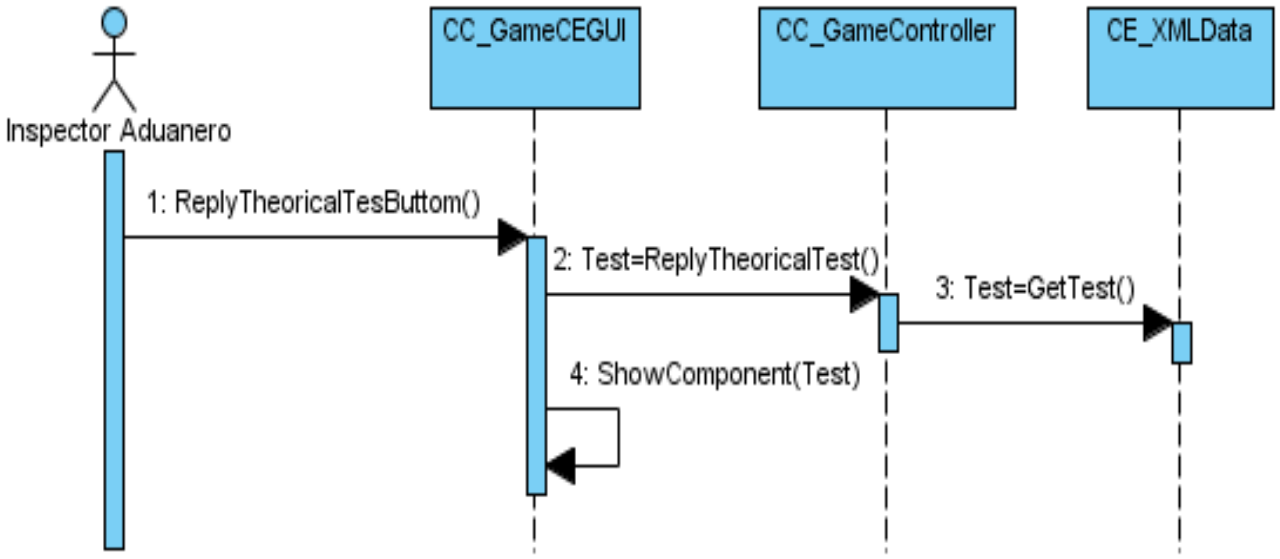

Figura 41: DCD "Responder test teórico"

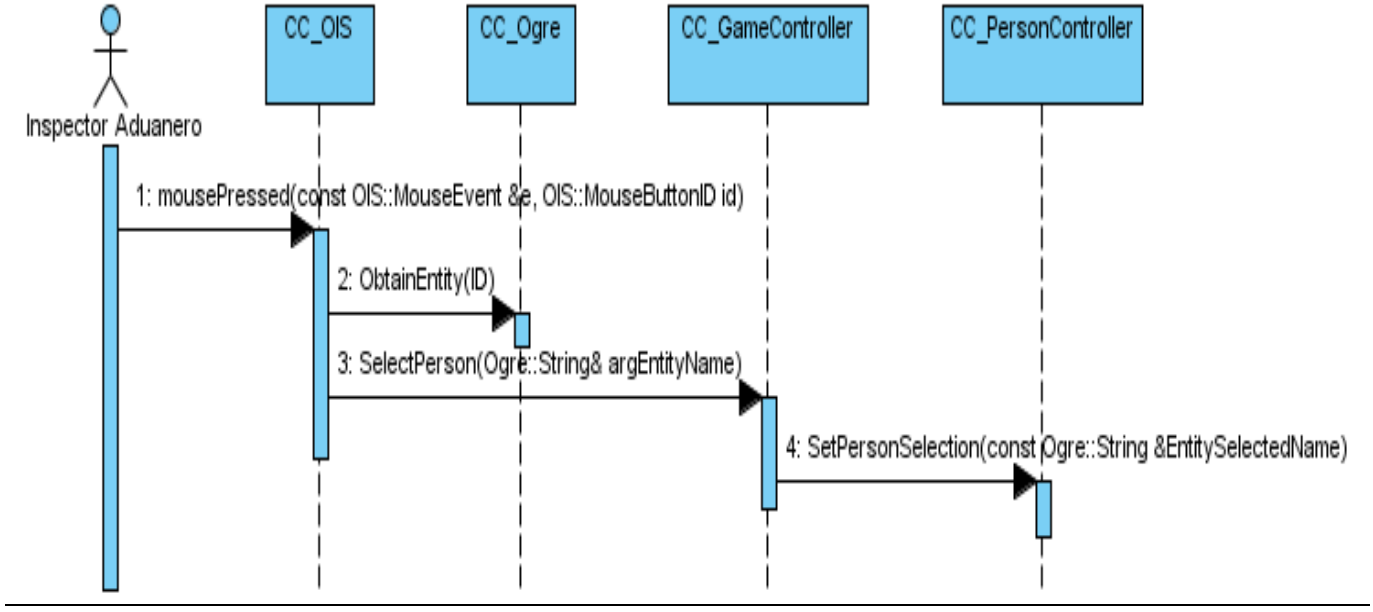

Figura 42: DCD "Seleccionar objeto"

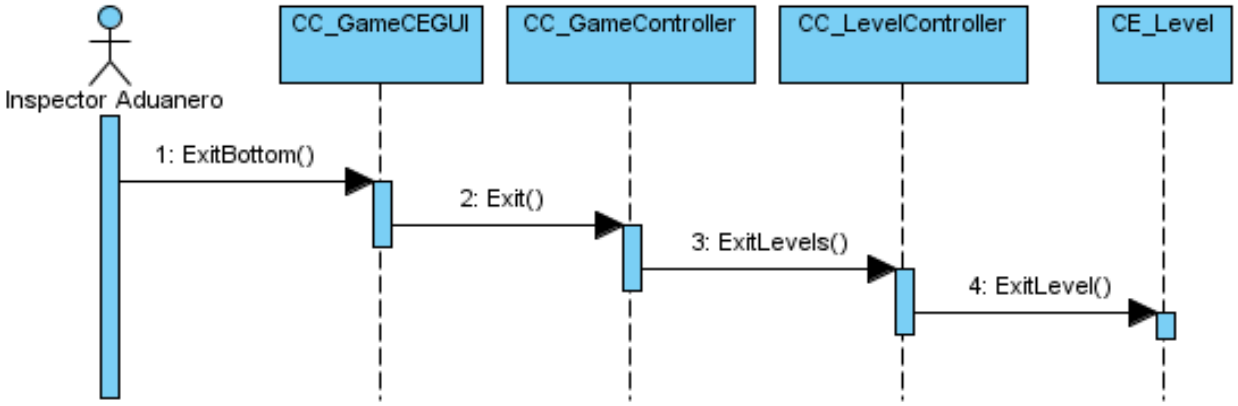

Figura 43: DCD "Terminar simulación (1)"

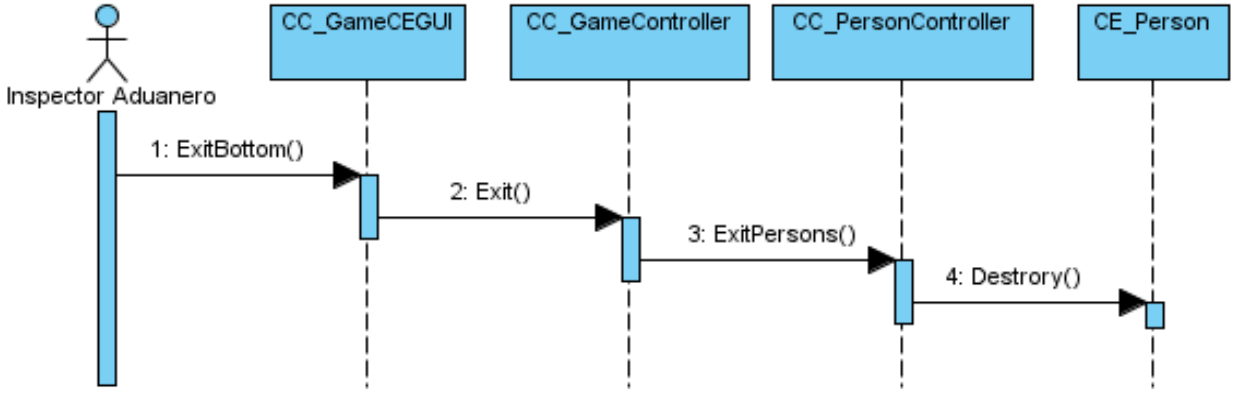

Figura 44: DCD "Terminar simulación (2)"

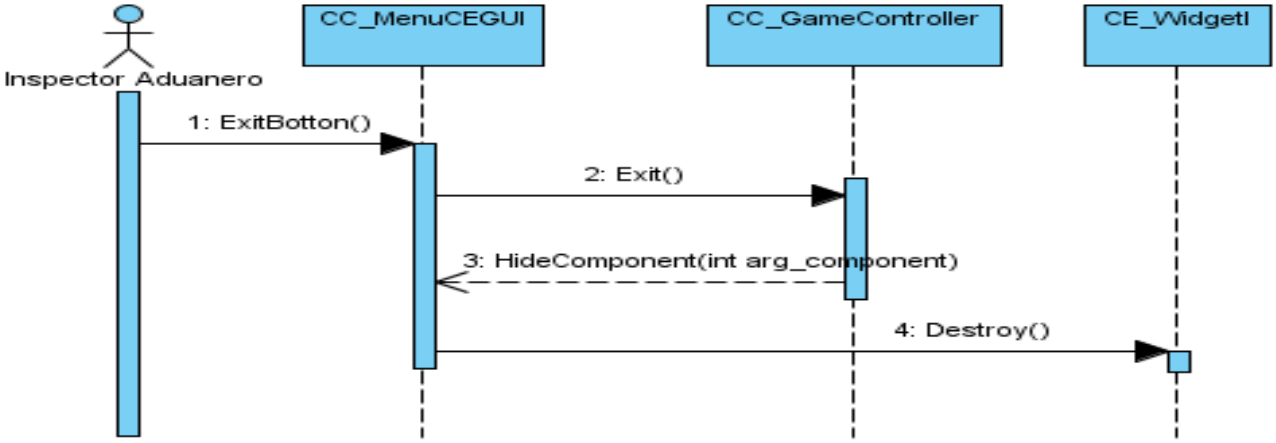

Figura 45: DCD "Terminar simulación (3)"

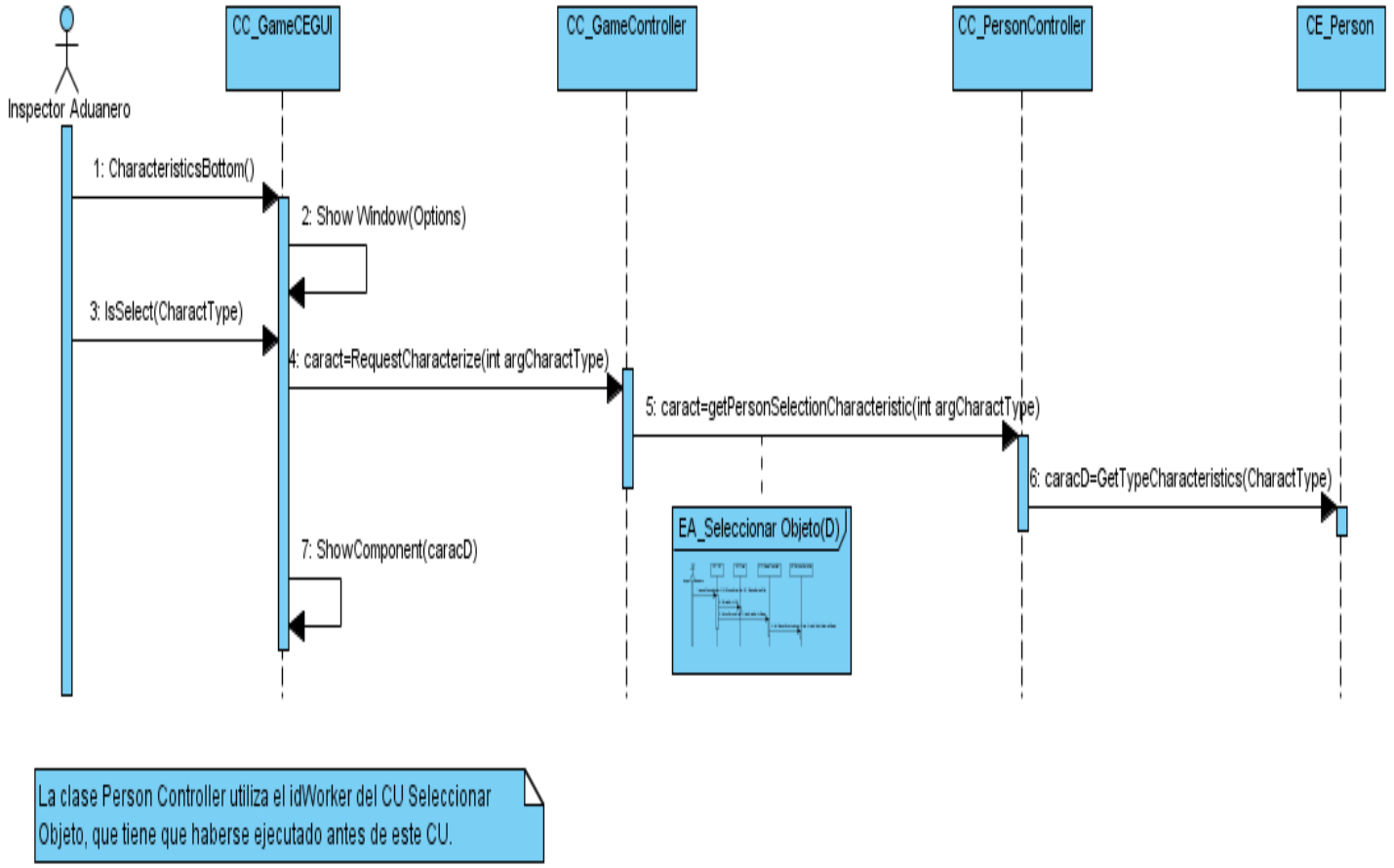

Figura 46: DCD "Ver características"

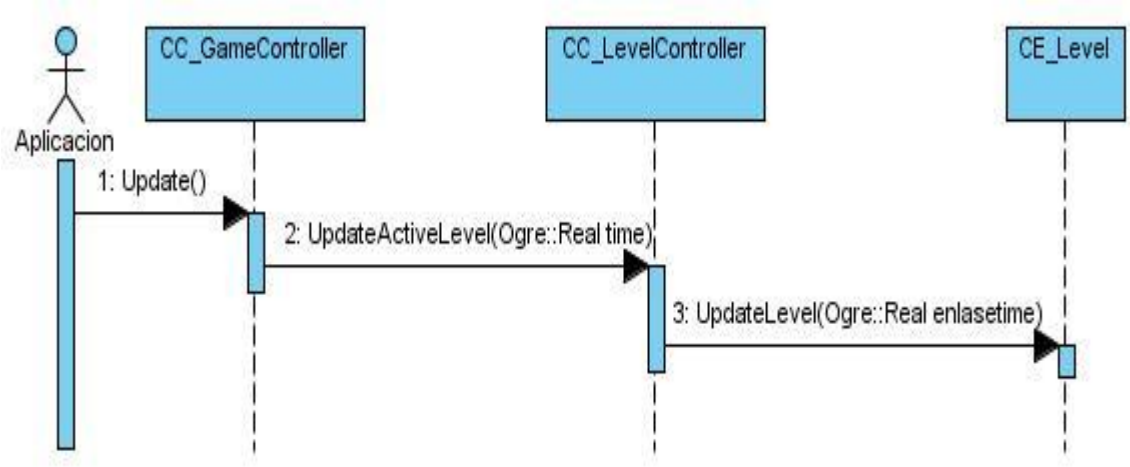

Figura 47: DCD "Actualizar niveles"

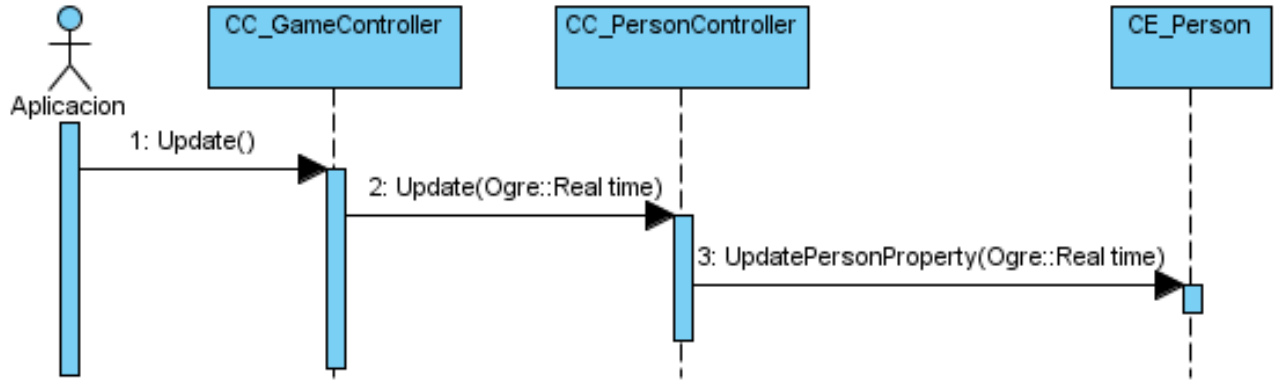

Figura 48: DCD "Actualizar personas"

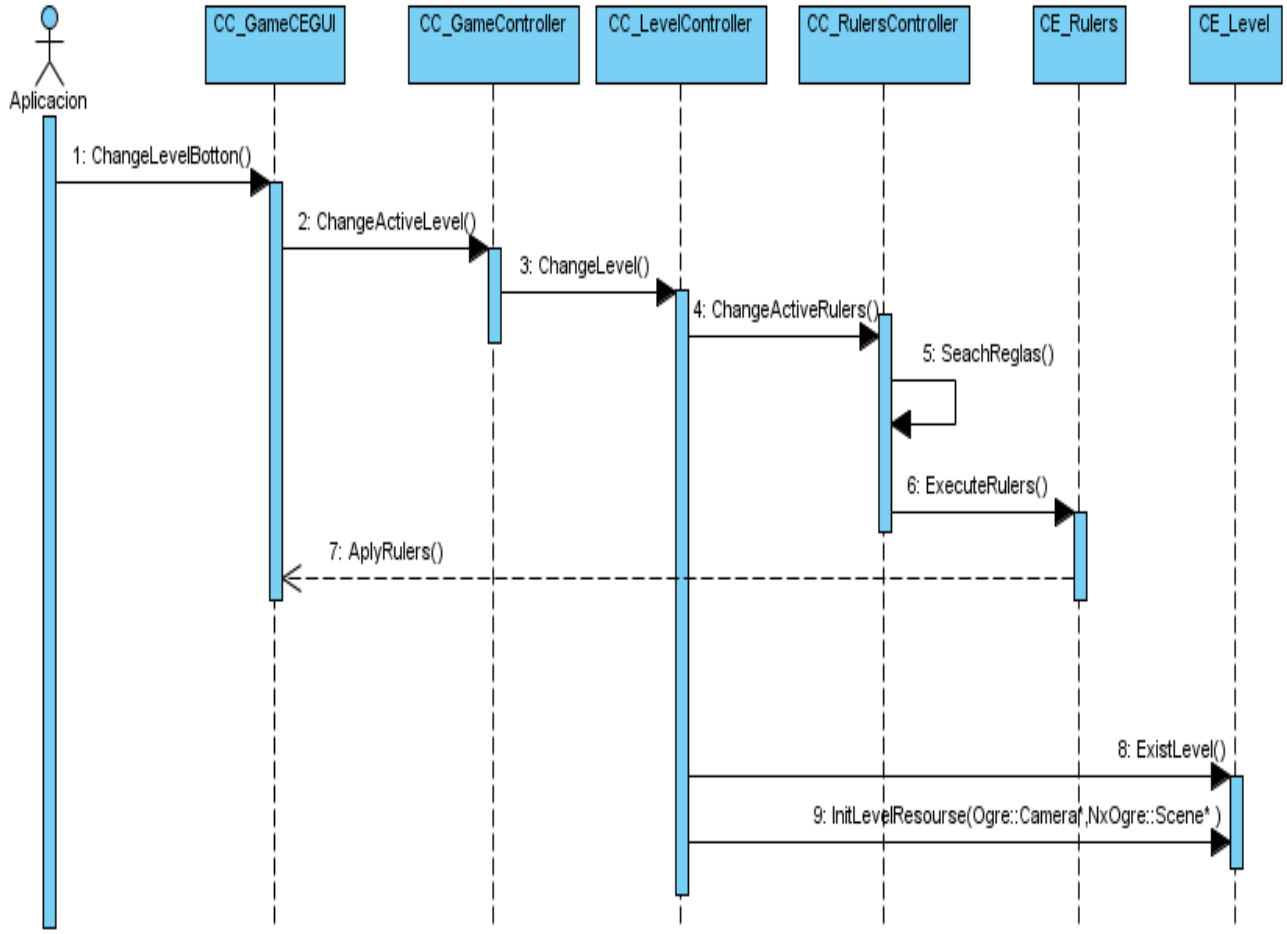

Figura 49: DCD "Gestionar niveles"

Capítulo 5: Implementación

En este capítulo se expondrán los diagramas correspondientes al flujo de implementación, como el diagrama de despliegue donde se reflejan los recursos que utiliza el módulo, también la estructura básica de cómo quedará conformada la implementación en cuanto a componentes físicos a partir del diseño de clases. Se presenta además, según el proyecto “Indicios Aduaneros” los estándares de codificación.

5.1 Estándar de codificación

Nombre de los ficheros:

Los nombres de los ficheros .h y .cpp utilizan las iniciales del nombre del proyecto (EA) y la primera letra de cada palabra en mayúscula.

ejemplo: EAGameController.cpp.

Clases:

Se utilizan las iniciales del nombre del proyecto (EA) al inicio del nombre de cada clase y la inicial de cada palabra que compone el nombre de la clase en mayúscula.

ejemplo: class EAGameController.

Punteros:

Los punteros a las clases se definen con el nombre de la clase a la que pertenecen, como exige el compilador y pk seguido del nombre del puntero.

ejemplo: ILevel * pkCurrentLevel.

Enumerados:

Se definen con el identificador del proyecto EA seguido del símbolo “_” y el nombre del enumerado.

ejemplo: EA_Salon_Finger.

Interfaces:

Se utiliza el indicador “I” para indicar que es una interfaz.
ejemplo: IRuler.

Listas e iteradores de la std:

Para los tipos de datos utilizados de la librería estándar de C++ (vector, map, multimap, etc.), se utiliza el indicador “EA”, con los sufijos List, Map y MultiMap según la estructura. Además el nombre lleva el tipo de dato a almacenar en la estructura en cuestión:
ejemplo: vector<Person*>kPersons
map<Levels_flag,ILevel*>kLevels

Declaración de variables:

Los nombres de las variables comienzan con un identificador del tipo de dato al que correspondan seguido de la inicial del tipo de dato y el nombre de la variable.

ejemplo: float fNombre.

bool bNombre.

int iNombre.

En caso de ser argumentos de algún método, se les antepone el prefijo “arg”.

ejemplo: funcionX(Tipo argNombre).

Los nombres de los atributos comienzan con “k” y si es un puntero a una clase se antepone “p”.

Métodos:

En el caso de los métodos, se les antepone el identificador del tipo de dato de devolución (void), y en caso de que no tenga, no se les antepone nada. Comienza con mayúscula al igual que cada una de las palabras que componen el nombre y no está separado por espacios. Los constructores y destructores, como lo exigen los compiladores, llevan el nombre de la clase.

ejemplo: void InitPersonResourse().

Constructor y destructor:

ejemplo: EAGameController();

~EAGameController();

Métodos de acceso a miembros:

Los métodos de acceso a miembros que se utilizan son los “GET” y los “SET”, comienzan con la palabra get o set en minúscula y el resto del nombre las palabras empiezan con mayúscula sin caparse por espacios.

ejemplo: getFlyDescription ()

SetTaskToWorker (int IndexOpcion)

5.2 Diagrama de despliegue

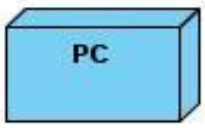

Figura 50: Diagrama de despliegue

5.3 Diagrama de componente

5.3.1 Subpaquete de componentes

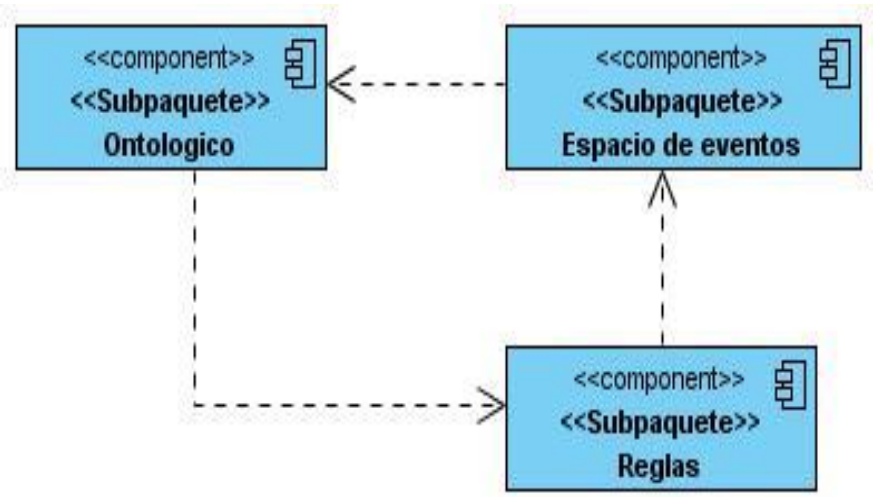

Figura 51: Subpaquete de componentes

5.3.2 Diagrama de componentes "Ontológico"

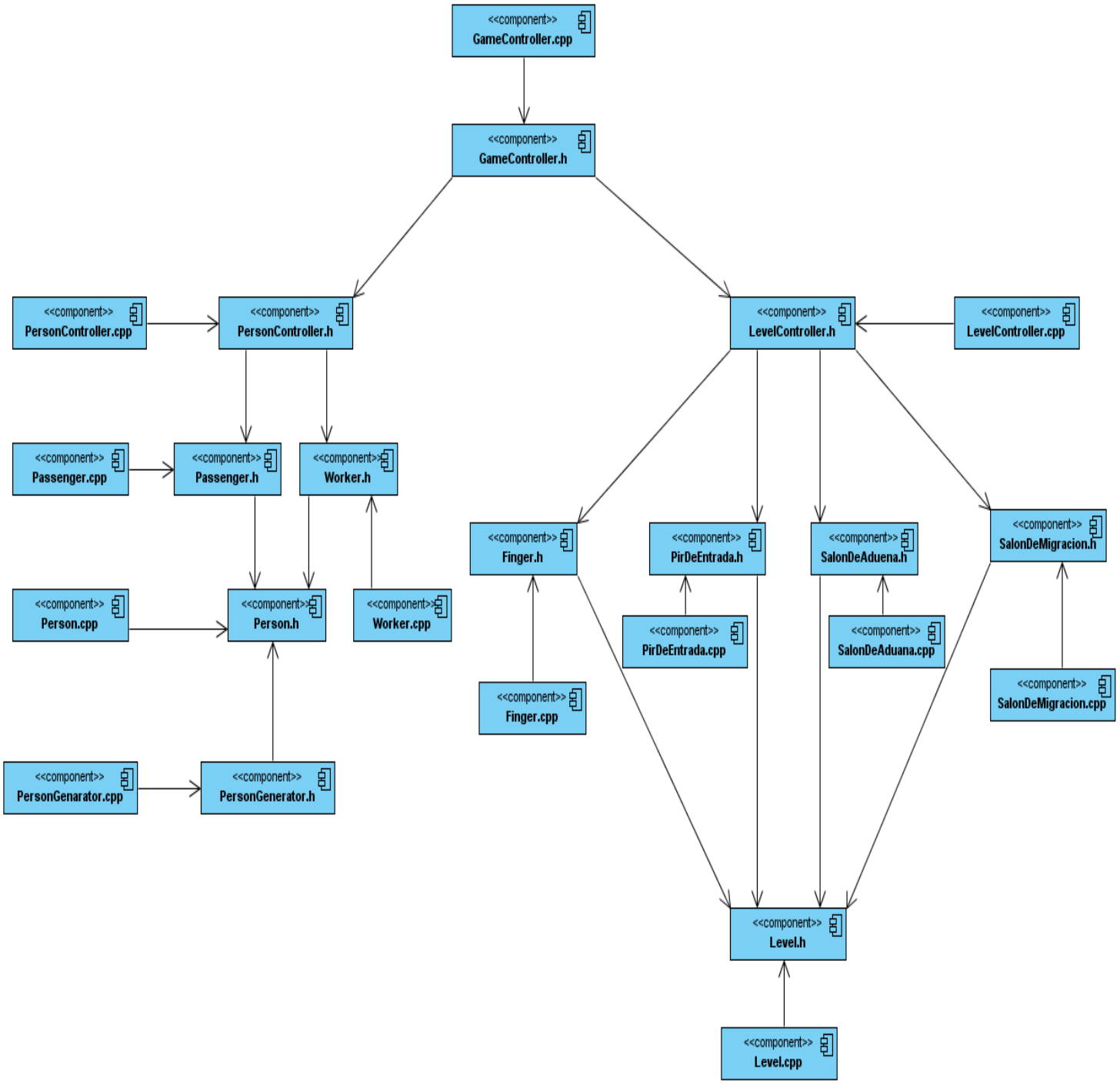

Figura 52: Diagrama de componentes "Ontológico"

5.3.3 Diagrama de componentes “Espacio de eventos”

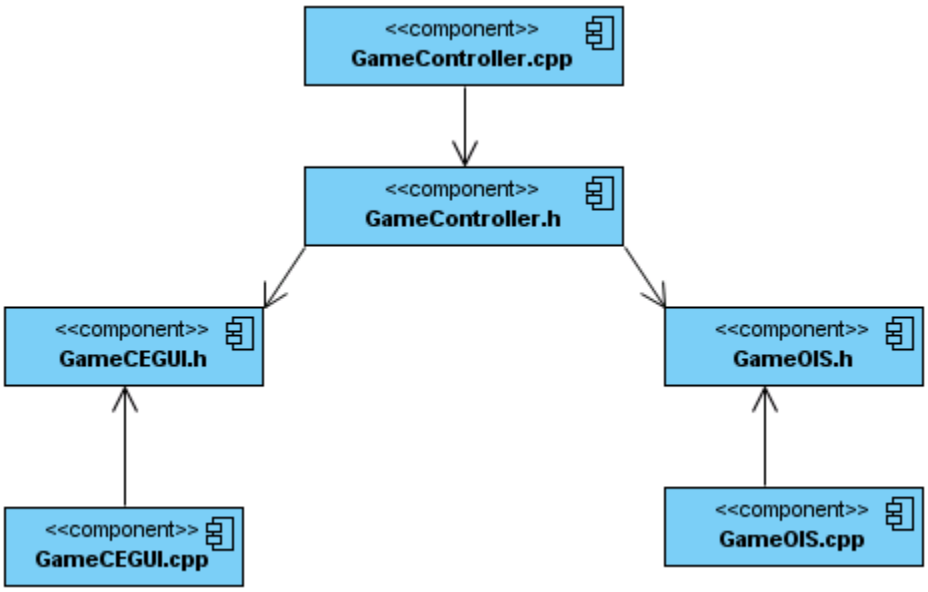

Figura 53: Diagrama de componentes “Espacio de eventos”

5.3.4 Diagrama de componentes “Reglas”

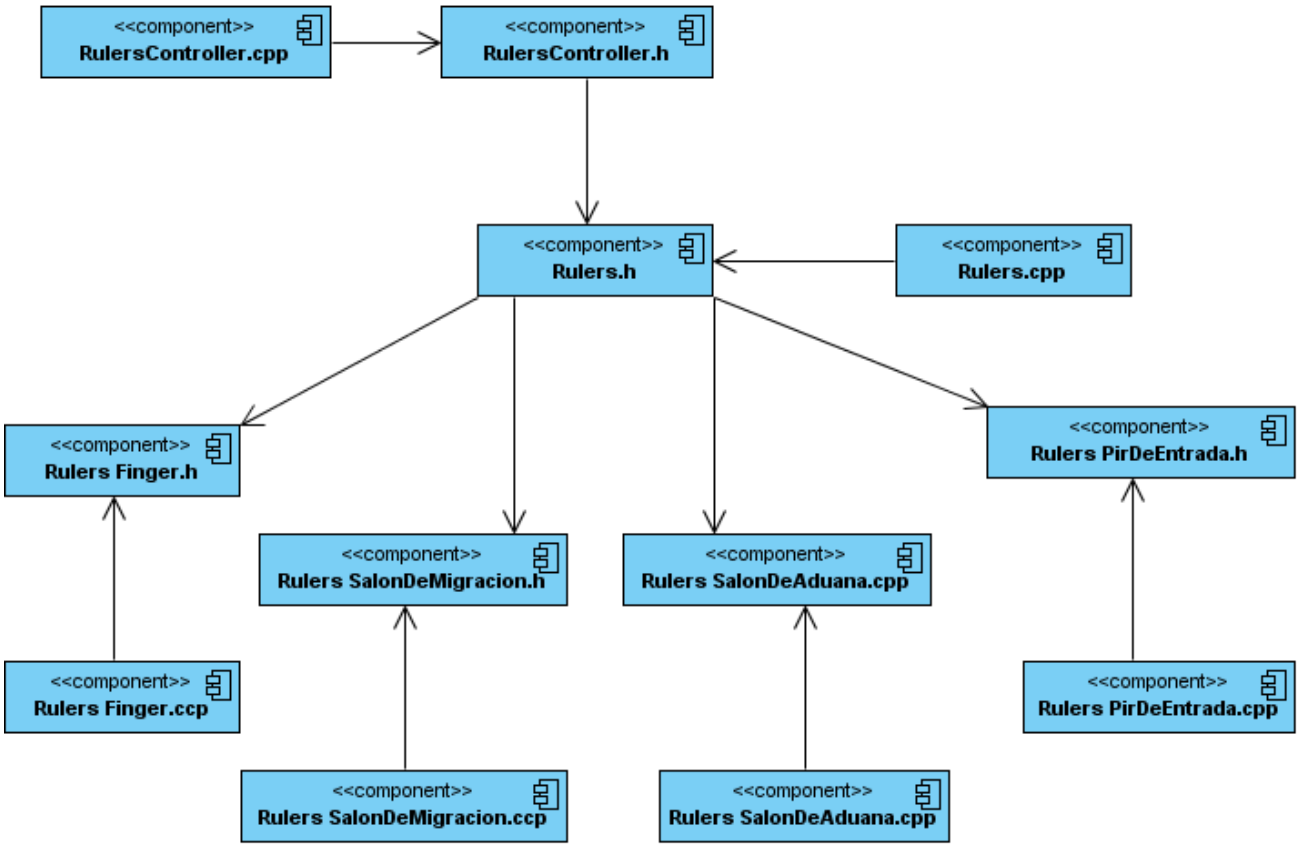

Figura 54: Diagrama de componentes “Reglas”

Conclusiones

Con el objetivo de satisfacer las necesidades planteadas por el cliente se realizó un estudio de las tendencias actuales en el diseño del motor lógico para juegos de simulación. Implementando este módulo de manera que permitiera centrar en un mismo subsistema toda la información referente a las reglas del juego, la lógica y las especificaciones de cada uno de los personajes y niveles. Con la finalización de la investigación y el desarrollo del módulo se da cumplimiento a los objetivos:

- ✓ Se realizó un subsistema reutilizable.
- ✓ Funcionamiento adecuado de la aplicación mediante la coordinación correcta de los módulos.
- ✓ Se elaboró una arquitectura que permite implementar nuevas reglas al módulo.

Recomendaciones

Se recomiendan los siguientes aspectos:

- ✓ Terminar la implementación del módulo siguiendo la arquitectura propuesta.
- ✓ Aplicar de la arquitectura de este módulo en otros proyectos.
- ✓ Implementar un método de evaluación para las respuestas del test teórico.

Referencias bibliográficas

BinSubaih, Ahmed, Maddock, Steve and Romano, Daniela. 2005. *Game Logic Portability.* Sheffield : s.n., 2005.

BinSubaih, Ahmed, Maddock, Steve and Romano, Daniela. 2004. *A Domain-Independent Multiplayer Architecture for Training.* Sheffield : s.n., 2004.

Buckland, MaT. 2005. *Programming Game AI by Example.* Texas : Wordware Editoria, 2005.

Carnegie Mellon University. 2009 . www.sei.cmu.edu. *www.sei.cmu.edu.* [Online] 2009 . [Cited: mayo 25, 2009.] http://www.sei.cmu.edu/architecture/definitions.html.

Chandrasekaran, B., Josephson, John R. and Benjamins, V. Richard. 1999. *Ontologies.* Amsterdam : s.n., 1999.

del Blanco Maraña, Alberto Carlos. 2007. *Diseño técnico shift videogame.* Madrid : s.n., 2007.

Departamento de Comunicaciones Comfamiliar Tuluá. 19. www.eltabloide.com.co. *www.eltabloide.com.co.* [Online] mayo 2009, 19. [Cited: mayo 26, 2009.] http://www.eltabloide.com.co/nuevo/index.php?option=com_content&view=category&layout=blog&id=97&Itemid=287&d909ec9ae35d8c826b52ae95dca4010b=17d714de2c05e3e58d4743a67ebf6707.

Duch i Gavaldà, Jordi and Tejedor Navarro, Heliodoro. *Diseño y Programación de Videojuegos.* Catalunya : s.n.

Duch i Gavaldà, Jordi and Tejedor Navarro, Heliodoro. *Introducción a los videojuegos.* Catalunya : uoc.

—. *Sonido, interacción y redes.* Catalunya : uoc.

EPSIG de la Universidad de Oviedo. 2008. shift.delblanco.es. *shift.delblanco.es.* [Online] 2008. [Cited: mayo 16, 2009.] http://shift.delblanco.es/develop/prog/index.html.

Hernández León, Rolando Alfredo and Coello González, Sayda. 2002. *El Paradigma Cuantitativo de la Investigación Científica.* Ciudad de la Habana : Eduniv, 2002.

Indice Latino. indicelatino.com. *indicelatino.com.* [Online] [Cited: abril 13, 2009.] http://indicelatino.com/juegos/.

Jacobson, I. and Booch, G. y Rumbaugh, J. 2000. *El Proceso Unificado de Desarrollo de software.* 2000.

Jacobson, I., Booch, G. Rumbaugh, J. and Addison-Wesley. 2000. *El Proceso Unificado de Desarrollo de software.* 2000.

Michael, David and Chen, Sande. 2006. *Serious Games.* 2006.

phpBB SEO. 2007. www.todojuegos.com. *www.todojuegos.com.* [Online] 2007. [Cited: abril 18, 2009.] http://www.todojuegos.com/ftopic30211.html.

Pressman, Roger and McGraw.Hill, . 2002. *Ingeniería de software. Un enfoque práctico.* Interamericana de España : s.n., 2002.

Pressman, Roger. 2002. *Ingeniería de software. Un enfoque práctico.* Interamericana de España : s.n., 2002.

Glosario de abreviaturas

AABB: *Axis Aligned Bounding Boxes*/Caja Alineada con los Ejes de Coordenadas.

AGR: Aduana General de la República.

BSP: *Binary Space Partition*/ Partición Binaria del Espacio.

CU: Casos de Uso

CUN: Caso de Uso del Negocio

CUS: Caso de Uso del Sistema

DCA: Diagrama de Clases del Análisis

DCD: Diagrama de Clases del Diseño

EDSAC: *Electronic Delay Storage Automatic Calculator*/Computadora Automática de Almacenamiento de Retraso Electrónico.

ENFA: Escuela Nacional de Formación Aduanero.

FSM: *Finite State Machine*/Máquinas de Estado.

IA: Inteligencia Artificial.

IAD: Inspector Aduanero.

OBB: *Oriented Bound Boxes*/Cajas Orientadas.

PI: Personal de Inmigración.

PPU: *Physical Processing Unit*/Unidad de Procesamiento Físico.

RUP: *Rational Unified Process*/Proceso Unificado Racional.

SDK: *Software Development Kit*/Kit de Desarrollo de Software.

UML: *Unified Modeling Language*/Lenguaje Unificado de Modelado.

XML: *Extensible Markup Language*/Lenguaje de Marcado Extensible.

Glosario de términos

Cachear: Registrar a personas sospechosas para detectar si porta objetos prohibidos que pueda llevar ocultos, por ejemplo armas, drogas etc.

CEGUI: Es un sistema de interfaz gráfica de usuario de la biblioteca de C + +. Está diseñado especialmente para satisfacer las necesidades de los videojuegos. Está diseñado para el usuario la flexibilidad de look-and-feel, además de ser adaptable a los deseos del usuario en herramientas y sistemas operativos.

CORBA IDL: *The CORBA Interface Definition Language/*El Lenguaje de definición de interfaces CORBA.

Discretización: Es la acción de dividir el todo en partes.

Espacio de eventos: Es la parte de la aplicación que recibe los acciones que proceden del los dispositivos de entrada teclado y mouse.

Finger: Pasillo por el cual transitan los pasajeros al bajar del avión.

Frame: Fotograma que conforma una animación, puede ser visto como el conjunto de imágenes que juntos conforman un video.

Framework: Es una estructura de soporte definida, mediante la cual otro proyecto puede ser organizado y desarrollado.

Handheld: Del idioma Inglés que significa llevar en la mano, describe a una computadora portátil para diversas aplicaciones, que puede ser llevada a cualquier parte mientras se utiliza.

Lógica: Es la parte del videojuego que controla la inteligencia de los avatares del videojuego y donde se programan todas las escenas, se definen e implementan las reglas del juego, la interacción entre los elementos y los comportamientos de todos los avatares.

Ludología: Es el campo de análisis de los videojuegos desde una perspectiva de Ciencias Sociales, Informática y Humanidades.

Módulo: es el sistema operativo, el núcleo de un videojuego. Se encarga de dirigir y coordinar cada uno de los aspectos tecnológicos ligados a él.

Ogre: Es un motor de renderizado 3D orientado a escenas, escrito en el lenguaje de programación C++.

OIS: Pretende ser una plataforma transversal, una solución simple para el uso de todo tipo de dispositivos de entrada (teclados, ratones, etc.) y dispositivos de información. Escrito en C + + usando patrones de diseño orientado a objetos.

Ontología: Este término se ocupa de la organización de la información a partir del estudio de sus propiedades, estructuras y sistemas, pueden clasificarse dentro de una jerarquía, y subdivididos de acuerdo a similitudes y diferencias. Por ello, trata de escribir o proponer las categorías y relaciones básicas de la existencia para definir las entidades y de qué tipo son. Las entidades comprenden los objetos, las personas, los conceptos, las ideas, las cosas, etc.

Pit de entrada: Es el salón donde se ejecuta el chequeo corporal y del equipaje de mano.

Round-robin: Método de selección de todos los elementos de un grupo de forma equitativa y con un orden racional.

Subsistema: Sistema que es parte de otro sistema, pero que puede tener determinado grado de independencia. Un sistema puede estar constituido por múltiples partes y subsistemas.

Salón de inmigración: Es el salón encargado de Verificar veracidad de la documentación (pasaporte, boleto de viaje).

Salón de la Aduana: Es el salón donde el personal de la Aduana realiza sus controles.

Teselación: Es una regularidad de figuras que cubre completamente una superficie plana que cumple con dos requisitos: que no queden huecos, que no se superpongan o traslapen las figuras.

***Weblog*:** Es un sitio web que actualiza constantemente y que recopila cronológicamente textos o artículos de uno o varios autores, donde el autor puede mostrar a los usuarios lo que crea pertinente.

Widget: Es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de widgets o *Widget Engine*.

Índice de tablas

Índice de figuras