Universidad de las Ciencias Informáticas

Facultad 5

Título: Sistema de gestión de diálogos para Entrenadores Aduaneros

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

Autor: José Enrique Gandón García

Tutor: Ing. Jaime González Campistruz

La Habana, Junio 2011

DECLARACIÓN DE AUTORÍA.

Declaro ser el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer el uso que estime pertinente de este trabajo.

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

Autor: José Enrique Gandón García

Tutor: Ing. Jaime González Campistruz

_____

_____

AGRADECIMIENTOS.

A mis padres, Águeda y José por su amor, apoyo y ayuda incondicional.

A Ángel y a Evert por ser más que amigos, hermanos.

A Miralys por su amor y comprensión.

A mi tutor Jaime por supervisar mi investigación.

A Alexey por estar presente cuando lo necesité.

A todos aquellos profesores y amigos que de una forma u otra hicieron posible este trabajo.

A todos ustedes, gracias.

DEDICATORIA.

A la memoria de mis abuelas: Ana y Antonia.

A mis padres.

RESUMEN

Los **juegos serios** son juegos creados con un fin no puramente de entretenimiento sino que son aplicaciones de tipo instructivo, como son los simuladores y entrenadores. Al igual que los **juegos de rol**, requieren en la mayoría de los casos de sistemas de diálogos que permitan interacciones entre los usuarios y la aplicación. Uno de los productos de la Universidad de las Ciencias Informáticas que exige el uso de este tipo de interacciones lo constituye el **Sistema de Entrenadores Aduaneros: Indicios Aduaneros,** destinado a la preparación de los inspectores aduaneros de recién incorporación a las filas de la **Aduana General de la República**. Se presentan dos módulos que tributan al producto principal**:** un **sistema de diálogo** y su correspondiente editor. El resultado de este trabajo ha constituido un primer paso en el desarrollo de este tipo de herramientas en la Universidad de las Ciencias Informáticas y en un futuro posibilitaría la ampliación de sus funciones para otros juegos, tanto de entrenamiento como de entretenimiento.

PALABRAS CLAVE

Sistema de diálogos, juegos serios, juegos de rol.

CONTENIDO

INTRODUCCIÓN

Actualmente la mayoría de los juegos basados en computadoras tienen gráficos 3D muy realistas, usan motores físicos complejos y la Inteligencia Artificial asociada demuestra un avanzado estudio del comportamiento humano y animal. Desde los inicios, en este tipo de entretenimiento, se evidenció el impacto de otra característica que ha primado en los mejores títulos del mercado: la comunicación entre agentes con el uso de sistemas de diálogos, ya sea entre el propio **personaje principal** (***P**layer **C**haracter* - ***PC***), manejado por el jugador y una **entidad autónoma** (***N**one–**P**layer **C**haracter* - ***NPC***) u otra posible combinación de ellos.

Existe diversidad de géneros de juegos como los de acción, deportes, etc. que no requieren de sistemas de diálogo complicados. Por el contrario, los **juegos de rol** (***R**ole **P**laying **G**ame* - ***RPG***), que son juegos donde los jugadores asumen roles de personajes o toman el control de uno o más avatares en un trasfondo ficticio, exigen sistemas de diálogos mejor definidos y más complejos. Aunque no clasifican entre los géneros de juegos reconocidos por la literatura científica, los **juegos serios** (***S**erious **G**ame* - ***SG***), que son juegos creados con un fin no puramente de entretenimiento sino con el propósito principal de enseñar y entrenar, también necesitan sistemas de diálogos complejos para poder cumplir con su función formadora.

En la Facultad 5 de la Universidad de las Ciencias Informáticas se desarrollan productos que exigen el uso de interacciones entre agentes en tiempo real en forma de diálogos. La motivación de este trabajo está enmarcada en la necesidad de disponer de una herramienta capaz de permitir estas interacciones.

Uno de los productos que exigen el uso de este tipo de interacciones es el **Sistema de Entrenadores Aduaneros: Indicios Aduaneros**. Este sistema es un videojuego destinado a la preparación de los inspectores aduaneros de recién incorporación a las filas de la **Aduana General de la República**. Al ser un videojuego de tipo **SG**, basa su contenido en un proceso real dado, con el objetivo de simularlo. En este caso es el proceso de detección de indicios por el cual se rigen los inspectores en el Aeropuerto Internacional “José Martí” de La Habana en Cuba, donde el objetivo principal es detectar los casos de ilegalidades que pretenden entrar en nuestro país (Campistruz, 2009).

El **Sistema de Entrenadores Aduaneros** no tiene actualmente entre sus funcionalidades, una que posibilite la comunicación entre los inspectores aduaneros que van a utilizar el sistema y la aplicación. Además, los productos que pudieran solventar en parte esta necesidad, no son libres y los que lo son no cumplen satisfactoriamente con la necesidad actual del proyecto.

Partiendo de la **situación problémica** antes expuesta, se plantea el siguiente **problema científico**: ¿Cómo desarrollar un sistema que permita la interacción mediante diálogos entre los usuarios del **Sistema de Entrenadores Aduaneros** y la propia aplicación?

Como **objeto de investigación** se tiene la gestión de diálogos en el desarrollo de juegos y de forma especial como **campo de acción**, los motores de diálogos en el desarrollo de juegos serios.

Con el propósito de brindarle solución al problema, se plantea como **objetivo general**, desarrollar un módulo para el **Sistema de Entrenadores Aduaneros** que permita la gestión de diálogos.

Las **tareas de investigación** que se deben realizar para dar cumplimiento al objetivo general planteado son:

- Analizar los videojuegos que utilicen diálogos en su trama.
- Comparar los diferentes tipos de diálogos usados en videojuegos.
- Analizar los diferentes programas que generan archivos con almacenamiento basado en XML para uso en juegos.
- Seleccionar el tipo de diálogo que más se ajuste al producto que se desarrolla.
- Seleccionar el tipo de almacenamiento y gestión de archivos XML más conveniente.
- Crear el editor XML.
- Crear el sistema de diálogo.
- Validar la propuesta.

Idea a defender:

Con el desarrollo del módulo para la gestión de diálogos en el **Sistema de Entrenadores Aduaneros**, se posibilitará la comunicación entre los inspectores aduaneros y la aplicación.

Para realizar esta investigación se utilizan los siguientes **métodos de investigación**:

- Métodos teóricos:
  - Histórico-Lógico: Permite estudiar el desarrollo y la evolución de las técnicas de diálogos en juegos para conocer su estado actual.
  - Analítico-Sintético: Permite recoger y resumir la información presente en los materiales consultados sobre el tema del desarrollo de máquinas de estado finitas y su uso en los motores de diálogo para utilizarlo en el desarrollo de este trabajo.
  - Deductivo-Inductivo: Permite seleccionar las alternativas idóneas para el diseño de los componentes sistema y editor de diálogos.

Estructura del trabajo:

El contenido de este trabajo está estructurado en tres capítulos de la siguiente manera:

- **Capítulo 1: Fundamentación teórica**.
  Este capítulo constituye una introducción a los conceptos básicos de mayor importancia para el tema principal de este trabajo y de aspectos relacionados con la problemática planteada en la introducción. Su explicación y análisis ayudará a un mejor entendimiento del mundo de los videojuegos y sus componentes, géneros y características.
- **Capítulo 2: Solución propuesta**.
  En este capítulo se describe la propuesta para dar solución a la necesidad actual de un sistema de diálogos para los Entrenadores Aduaneros. Aquí se exponen las herramientas, el lenguaje y la metodología a seguir para la creación del sistema.
- **Capítulo 3: Análisis y diseño del sistema**.
  En este capítulo se presenta el modelo del negocio, los actores que interactuarán con el sistema, los requisitos funcionales y no funcionales que debe tener éste y se definen los casos de uso. Además se presentan los diagramas de casos de uso para representar la interacción de los actores con el sistema y se realiza una descripción textual de estas interacciones.

  Además se presentan los diagramas de clases del diseño y los diagramas de componentes.

CAPÍTULO 1: FUDAMENTACIÓN TEÓRICA.

INTRODUCCIÓN.

Este capítulo constituye una introducción a los conceptos básicos de mayor importancia para el tema principal de este trabajo y de aspectos relacionados con la problemática planteada en la introducción. Su explicación y análisis ayudará a un mejor entendimiento del mundo de los videojuegos y sus componentes, géneros y características.

1.1. VIDEOJUEGOS.

Los videojuegos han formado parte del entretenimiento humano desde inicios de los años 70 y por ello muchas definiciones sobre el tema han ido saliendo gradualmente a la luz y en literatura de diversa naturaleza. Aquí se listan algunas de las más reconocidas por la literatura de divulgación:

- *Un videojuego o juego de video es un software informático creado para el entretenimiento en general y basado en la interacción entre una o varias personas y un aparato electrónico que ejecuta dicho videojuego* (USPTO, 1995).
- *Entendemos por videojuegos todo tipo de juego digital interactivo, con independencia de su soporte* (Graells, 2001).

El éxito de un videojuego - independientemente de su complejidad, de su fin y su género - se juzga en primera instancia por la calidad de su diseño. Son tres los aspectos que lo determinan (Brusk, y otros, 2007):

- La motivación que logra el diseño del juego en el jugador.
- El nivel de inmersión que provee al jugador.
- La integración de las **componentes**.

Las componentes son las secciones en las que están divididos los videojuegos. Cada una tiene su propósito y solamente integradas constituyen esta forma de entretenimiento.

Seguidamente se listan las principales componentes de un videojuego.

- **Gráficos tridimensionales**: Es la representación visual de textos, imágenes y colores aplicada a una superficie bidimensional (**2D**) o a un objeto tridimensional (**3D**).
- **Sonido y música**: El sonido es cualquier fenómeno que involucre la propagación en forma de ondas elásticas. (Simón, 2005) Al organizar sensible y lógicamente una combinación coherente de sonidos y silencios se produce la música (Ulrich, 1985).
- ***Jugabilidad***: Término empleado en el diseño y análisis de juegos que describe la calidad del juego teniendo en cuenta sus reglas de funcionamiento y su diseño como juego. Se refiere a todas las experiencias de un jugador durante la interacción con sistemas de juegos.
- **Historia**: La historia en los videojuegos se refiere a la trama en la cual se desarrollan los acontecimientos donde uno o varios personajes se desenvuelven.
- **Diálogos**: El diálogo puede ser descrito como un término para agentes que intercambian y coordinan información, construyen cohesión social y logran entendimiento mutuo (Alwood, 1992).

La forma de utilización de estas componentes define en gran medida que tipo de videojuego se va a realizar. A estos tipos de videojuegos se les conoce como géneros.

Los principales géneros identificados en la literatura de divulgación son:

- Acción
- Acción-Aventura
- Simulación
- Estrategia
- Simulación de Vehículos
- Juegos de Rol
- Otros géneros

En los últimos tiempos, y bajo la clasificación de otros géneros aparecen los llamados Juegos Serios. Este tipo de juego no se considera en las clasificaciones actuales de

videojuegos que la literatura de divulgación identifica y en este sentido se incluyen: juegos de arte, juegos educativos con fines de aprendizaje o cognitivos o de producción de conocimiento (Derrberry, 2005). De tal relevancia se consideran sus aportes que se les conoce también como *juegos de aprendizaje*.

Para crear un videojuego con un ambiente creíble, para que "*los personajes provean la ilusión de vida*" (Bates, 1994), éstos deben poseer cualidades inherentes a los humanos como emociones, personalidad, la habilidad de comunicarse en lenguaje natural, comportamiento adaptable y capacidades de aprendizaje (Gratch, y otros, 2002).

En los videojuegos los jugadores interactúan con estos personajes y en la mayoría de ellos se realiza a través de un de las **componentes**: los **diálogos**.

1.2. DIÁLOGO.

Como se refirió en la sección anterior, el diálogo puede ser descrito como un término para agentes que intercambian y coordinan información, construyen cohesión social y logran entendimiento mutuo (Alwood, 1992) y en este trabajo se interpreta de forma tal que pueda ser aplicado al contexto de **diálogo en videojuegos**.

Al estar los videojuegos constituidos por acciones y acontecimientos - en las que el jugador debe participar – así como búsquedas o acertijos - que el jugador debe solucionar - una razón principal para comunicarse con los personajes lo constituye la colocación de un indicio de qué hacer después así como también la motivación para ello. En la mayoría de los videojuegos estos diálogos son puramente funcionales y mostrados al jugador a través de interfaces de diálogo estático (Brusk, 2006).

Los géneros de videojuegos que utilizan los diálogos de la manera descrita anteriormente son los **juegos de rol** y los **juegos serios**. Aunque existen diversos puntos donde coinciden, existen dos que rigen su denominación, donde se separan completamente y son: los destinatarios o público al cual está dirigido y el objetivo que persigue. En estos dos aspectos está la gran brecha entre estos géneros.

JUEGOS DE ROL.

Los **juegos de rol** constituyen una amplia familia de juegos donde los jugadores asumen roles de personajes o toman el control de uno o más avatares, en un trasfondo ficticio. Las acciones tomadas dentro del juego tienen éxito o fracasan según reglas y líneas directivas.

Al tener estos juegos como público a adolescentes y adultos, las diferentes empresas desarrolladoras de éstos han logrado de manera eficaz ajustar el juego a la capacidad de cada cual. Esta medida se puede encontrar en los niveles de dificultad de acertijos, tareas y características de los avatares en el juego. Además, las componentes tienen que acoplar perfectamente para que sea adquirido el juego (como se ha dicho anteriormente, por la gran cantidad de títulos presentes en el mercado) pues el entretenimiento es su principal función.

La historia en estos juegos está basada en la exploración del mundo, como se puede ver en la **Figura 1**, donde cada capítulo se desarrolla en diferentes localidades. Usualmente se permite que los jugadores regresen a las zonas anteriormente visitadas del mundo por una necesidad de la historia o solo por recreación del jugador.

Figura1: The Elder Scrolls IV: Oblivion.

En lo referente a las misiones o tareas dentro del juego, siempre un grupo de éstas son las definitivas para terminar el juego y se comportan de forma lineal. Las demás son opcionales y se usan para dotar al jugador de más habilidad y alargar la estancia del usuario en el juego.

JUEGOS SERIOS.

Los **juegos serios** son juegos creados con un fin no puramente de entretenimiento, sino con el propósito principal de enseñar y entrenar, ya sea divulgando, investigando o simulando, como ya se explicó anteriormente. Se refiere a productos utilizados principalmente por instituciones educacionales, médicas, militares, religiosas, políticas, etc.

El término "juego serio" fue acuñado antes de la introducción de las consolas y las computadoras para su uso en el entretenimiento. Fue Clark Abt en su libro de 1970 titulado **Serious Games** el primero en utilizarlo y discutir sobre él.

Su definición del término puede ser aún aplicada hoy en día:

> *Reducido a su esencia formal, un juego es una actividad entre dos o más decisores que toman las decisiones independientemente tratando de lograr sus objetivos en algún contexto limitado. Una definición más convencional sería que un juego es un contexto con reglas entre adversarios intentando ganar objetivos. Estamos de acuerdo con que los juegos serios en esencia, tienen un explícito y meditado propósito educativo y no están intencionados para la diversión.* – Clark Abt.

Estos juegos son desarrollados para un público en específico, pues su propósito en general es educar, no importa el campo donde se vaya a utilizar. Además, no todos los componentes necesitan ser desarrollados al máximo, por ejemplo, un juego serio para entrenamiento militar no requiere de sonidos y música de alta calidad, ni historia envolvente, ni diálogos muy conformados; por el contrario, uno para entrenamiento médico quizás requiera alta calidad de sonidos u otro para entrenamiento en la actividad aduanera si requiera de diálogos bien organizados.

De una forma u otra, en todos los videojuegos están presentes los diálogos. Por lo cual se pueden diferenciar tres tipos principales de interacción verbal potencial. Puede ser en forma de texto o hablado (Brusk, 2006):

- **Meta-comandos**: Constituye la forma más sencilla y frecuente en que se pueden encontrar los diálogos. Son comandos y funciones de alto nivel que están presentes a la hora de llevar al jugador a salvar o cargar su trayectoria en el juego y acceder a las opciones o salir. Este tipo de diálogo es un componente obligado de todos los tipos o géneros de juegos.
- **Diálogo funcional o diálogo participativo**: Son los diálogos entre el PC y un NPC en el juego. Proveen al jugador de historia de fondo, búsquedas, consejos y demás. La mayoría de los juegos de aventura y de rol tienen esta característica. Estos diálogos son usualmente exhibidos utilizando una interface gráfica estática, impidiéndole al jugador tomar iniciativa.
- **Diálogo social**: Define conversaciones en el juego entre PCs así como también las conversaciones PC-NPC. El propósito principal de estos diálogos es desarrollar y sostener relaciones y afectar emotivamente a los otros participantes.

De estos tipos de interacción, el **diálogo funcional o diálogo participativo** es el que más contacto tiene en todo momento con el jugador y es el que más información transmite. Por esto se necesita de un sistema adjunto al videojuego que gestione que diálogo mostrar en dependencia de las situaciones que se presenten en un momento dado de la trama y de la acción del jugador en esta. Para este fin se utilizan los **sistemas de diálogo**.

1.3. SISTEMAS DE DIÁLOGOS.

Los aspectos lingüísticos cobran un interés creciente en la comunidad científica, en especial en las Ciencias Informáticas, identificándose un área del conocimiento llamada Lingüística Computacional. Los diálogos, como modelo natural de comunicación humana, son de interés de la Lingüística y la Comunicación. En el caso que ocupa a este proyecto, el interés está dirigido a los denominados modelos de diálogos, lo cual constituye una parte de la Lingüística Computacional (Brusk, y otros, 2009).

La literatura científica en este último tema no abunda. Esto se comprende por la necesidad lógica de los productores de proteger las bases teóricas y de diseño de sus productos en un mercado de creciente demanda y cada vez más competitivo. Una entidad de autores asociados en **DiGRA** (**Di**gital **G**ames **R**esearch **A**ssociation) ha publicado diversos artículos de corte científico en relación a patrones de diseño (Brusk, y otros, 2009).

Desde el punto de vista computacional, se entiende por sistemas de diálogo a aquel sistema que pretende la comunicación con el humano. Estos sistemas emplean textos, gráfica, voz, gestos y otras formas de comunicación que se integran, tanto en el canal de entrada como el de salida, de una manera coherente y a través del lenguaje natural (Jufarsky, y otros, 2009).

La elección de la tecnología para la gestión de diálogos depende de factores como (Brusk, 2006):

- **Requerimientos de diálogo**: Que características del diálogo debe manejar el sistema para responder a la necesidad de la aplicación.
- **Entorno de la aplicación**: En qué contexto va a ser utilizado el sistema de diálogos.
- **Simplicidad**: Se debe escoger una tecnología fácil de utilizar pero capaz de manejar las tareas.

Existen cuatro formas principales para clasificar los sistemas de diálogo (Cassell, y otros, 2000) (Mc Tear, 2002):

- **Basado en estados finitos**: Pueden ser modelados como un grafo formado por nodos representando estados y con arcos entre estos que denotan transiciones. El usuario pasará de un estado a otro cuando se cumpla la transición correspondiente.
- **Basado en formas**: Se utiliza una especie de formulario donde se guarda la información seleccionada o escrita por el usuario; a partir de ella y utilizando un algoritmo de control de diálogo se determina la siguiente acción del sistema. Pueden ser modelados varios estados para una misma entrada, así como manipular las entradas en diferente orden.
- **Basado en plan**: El sistema y el usuario colaboran para resolver una tarea específica en un entorno estable, o sea, en el que las condiciones no cambian.
- **Basado en agentes**: Es utilizado en mundos virtuales donde el ambiente cambia dinámicamente y por ende las condiciones. En este caso la conversación requiere de un modelo de agente con la suficiente habilidad para re-planificarla.

Se pudo corroborar que, en el caso **basado en formas**, existen dos explicaciones. Esto se debe a que es muy difícil clasificar los sistemas de diálogos tomando en cuenta todas las variaciones que las compañías encargadas de desarrollarlos, según el producto, elijan.

Estas formas de diálogo pueden combinarse entre sí, todo dependerá del tipo de juego que se desee desarrollar, del contenido específico y de la complejidad que se le desee proporcionar.

Más allá de los tipos de diálogos, otra forma de clasificarlos es refiriéndose a las **funciones de control de diálogo** (Bunt, 1994) **o patrones de diseño de jugadas**, que no son más que el uso de técnicas para controlar las conversaciones y hacerlas creíbles. Seguidamente se explican (Björk, y otros, 2005):

- **Toma de turnos**: Es un control de elementos de diálogo básico que tiene que ver con cómo se distribuyen y organizan los movimientos de foco en una conversación sin cambiar el aspecto comunicativo de la interacción.
- **Procesamiento incremental del diálogo:** Usado en las conversaciones entre jugadores donde la interrupción es posible. Resulta importante cuando la contribución del que está leyendo el texto del que escribe es esencial.
- **Procesamiento de diálogo basado en secciones**: Alternativa de la anterior donde el diálogo es procesado por secciones para que los mensajes siempre sean entregados.
- **Diálogo de iniciativa solitaria**: Cuando un diálogo solo puede ser iniciado por uno de los agentes.
- **Diálogo de iniciativa mixta**: Cuando un diálogo puede ser iniciado por cualquiera de los agentes.
- **Información retroactiva básica**: El agente en posesión del turno espera confirmación de que los demás participantes han oído y comprendido el mensaje.
- **Interrupción**: Un agente que está oyendo al orador puede interrumpirlo convirtiéndose éste en orador.

Estas funciones de control de diálogo pueden coexistir en un mismo sistema y hasta combinarse entre sí, todo dependerá - al igual que en las clasificaciones de los tipos de diálogo - del tipo de juego que se desee desarrollar, del contenido a tratar y de la complejidad que se le desee proporcionar.

Diversos son los sistemas de diálogo en el mundo de los juegos, tanto así que pueden existir diferentes para los mismos tipos de éstos en una misma compañía productora.

En la actualidad un ejemplo lo constituye **BioWare Corp.**, compañía subsidiaria de la gigante **Electronic Arts** y dueña de varios juegos **RPG** de renombre como **Baldur's Gate**, **Neverwinter Nights**, **Jade Empire**, **Mass Effect**, **Dragon Age: Origins**, **Dragon Age 2** y **Mass Effect 2**. Este último constituyó uno de los juegos más vendidos de los últimos años (Electronic Arts News, 2010).

BioWare Corp. y otras compañías que se asocian a ella, han creado diferentes sistemas de diálogo para los juegos mencionados y por informaciones dadas por la compañía, posiblemente el último de estos sistemas - creado específicamente para **Mass Effect 2** y cuyo protagonista se puede ver en la **Figura 2**, se implementará en todos los futuros juegos de la firma. Prueba de ello es la reciente aparición en el mercado de la secuela del juego **Dragon Age**, titulado **Dragon Age 2** (Electronic Arts News, 2010).

Figura 2: Diálogo en Mass Effect 2.

El sistema de diálogos creado para **Mass Effect 2**, por difícil que parezca, es uno de los más sencillos de los que se han creado, pues utiliza la creación de diálogos **basado en**

estados finitos y de los patrones de diseños explicados anteriormente utilizan los más básicos y sencillos, **toma de turnos** y **diálogo de iniciativa solitaria**.

Con este importante ejemplo se puede llegar a la conclusión de que un buen sistema de diálogos no tiene que ser del tipo más complicado ni tener implementados varios patrones de diseño, lo importante es un equipo de trabajo que desarrolle diálogos llamativos, situaciones creíbles y temas interesantes.

Para que los editores y lingüistas encargados de los diálogos puedan ingresar los textos de los diálogos en el sistema, crear las condiciones y variables, es necesaria la utilización de un **editor de diálogos**, un programa que almacena los diálogos en un archivo que después se cargará en el videojuego ante la situación correspondiente.

1.4. EDITORES DE DIÁLOGOS.

Los sistemas de diálogo de las grandes compañías se integran junto a los motores de juegos y en muchas ocasiones no tienen siquiera nombre propio, solamente son un módulo que se puede cambiar a gusto de la empresa.

Junto a los sistemas de diálogos se encuentra el **editor de diálogo**, que no es más que el ambiente donde se crean las conversaciones y se adicionan los resultados y condiciones de cada interacción. Los que se detallan a continuación son editores que simplemente generan un archivo con la conversación y sus características, generalmente en formato XML:

- **Chat Mapper**: Potente editor de diálogos creado por **UrbanBrainStudios** con una interfaz muy amena basada en nodos. Para poder exportar a ficheros, comúnmente utilizados por los juegos, es necesario registrarlo, por lo que no es libre. Ver la **Figura 3**.
- **Adonthell Dialog Editor**: Editor libre también basado en nodos con una interfaz un poco arcaica que responde a un juego de rol gratuito con editor de mundos y motor gráfico propios.
- **XML Dialog-Editor**: Editor libre creado por el grupo **Anheledir** bajo permiso de **Microsoft** utilizando la **Licencia Recíproca de Microsoft**. No está basada en nodos sino en dos vistas, lista y forma de árbol.

- **Visual3D's Conversation Editor**: Parte del motor **Visual3D Game Engine**, propiedad de **Realmware Corporation** utiliza la vista en forma de árbol. Ver la **Figura 4**.

Figura 3: Chat Mapper.

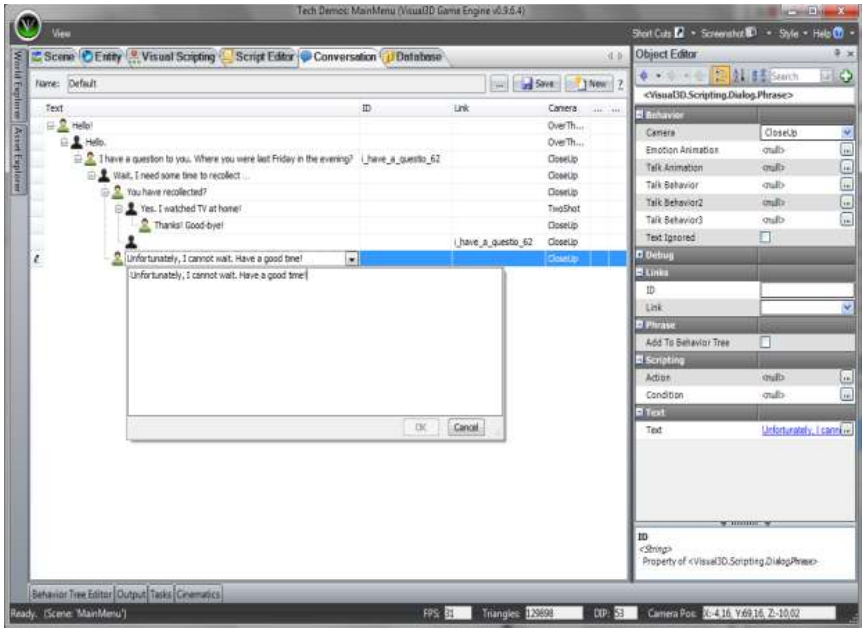

Figura 4: Visual'3Ds Conversation Editor.

Como se puntualizó anteriormente, la mayoría de los editores de diálogos almacenan los datos y textos de los diálogos en archivos con formato XML. Esto se debe principalmente a la forma de almacenamiento que brinda éste.

1.5. XML.

El Lenguaje de Marcado Extensible (**XML** – **E**xtensible **M**arkup **L**anguage) es un conjunto de reglas para la codificación de documentos (W3C). Es una simplificación y adaptación del **SGML**[1] y fue desarrollado por el World Wide Web Consortium (**W3C**).

El diseño de XML permite el énfasis en la simplicidad, generalidad y usabilidad sobre Internet (W3C). Es un formato de datos textual con soporte vía **Unicode**[2] para los demás lenguajes. Aunque está centrado en los documentos, es ampliamente utilizado en la representación de estructuras de datos arbitrarios, por ejemplo, en los servicios web.

Existen muchas interfaces de programación que los desarrolladores pueden utilizar para acceder a datos XML y varios sistemas que ayudan en la definición de lenguajes basados en XML. Se han desarrollado cientos de lenguajes basados en XML (XML Coverpages), incluyendo **RSS**, **Atom**, **SOAP** y **XHTML**.

Los formatos basados en XML se han convertido en los más usados por parte de las herramientas de escritorio, incluyendo Microsoft Office (Office Open XML), OpenOffice.org (OpenDocument) y el iWork de Apple (Apple).

Otro de los formatos basados en XML es el **Statechart**[3] **XML** (**SCXML**), ampliamente utilizado en juegos de rol y juegos serios.

SCXMLpuede ser descrito como un intento de crear gráficos de estados Harel en XML. David Harel desarrolló sus gráficos de estados como una notación gráfica para especificar sistemas reactivos con gran detalle. En su forma más simple, un gráfico de estados es simplemente una máquina de estado finita, donde las transiciones entre estados se desencadenan por los acontecimientos que aparecen en una cola de estos.

[1] **SGML** (**S**tandard **G**eneralized **M**arkup **L**anguage [ISO 8879:1986]) – El Lenguaje de Marcado Generalizado Estándar es un estándar tecnológico ISO para definir lenguajes de marcado generalizado para documentos. Se utiliza para organizar y etiquetar éstos.
[2] **Unicode** – Estándar de la industria computacional para la representación y manejo de texto expresado in la mayoría de los sistemas de escritura.
[3] **Statechart** – Gráfico de estados.

Cualquier gráfico de estados puede ser traducido a un documento escrito en la sintaxis basada en XML lineal de SCXML(Brusk, y otros, 2007).

Existen **API**s[4] para acceder a documentos XML han sido desarrolladas y utilizadas, y varias han sido estandarizadas.

1.6. INTERFACES DE PROGRAMACIÓN.

Las APIs para el procesamiento de XML se pueden incluir en estas categorías(W3C, 2011):

- *Stream-oriented* APIs: Accesible desde un lenguaje de programación, por ejemplo **SAX** y **StAX**.
- *Tree-traversal* APIs: Accesible desde un lenguaje de programación, por ejemplo **DOM**.
- *XML data binding*: Conversión automática de un documento XML a un objeto en memoria.
- Lenguajes declarativos de transformación: **XSLT** y **XQuery**.

De las interfaces de programación de aplicaciones nombradas en la lista anterior, SAX y DOM son las más utilizadas en los **IDE**[5] para leer datos desde archivos XML. A continuación se explican con más profundidad.

1.6.1. API SIMPLE PARA XML (SAX).

Es un *parser*[6] API de acceso en serie para XML. Provee de un mecanismo para leer datos de un documento XML y es una popular alternativa al DOM. Originalmente fue una API únicamente para el lenguaje de programación JAVA. Existen versiones de SAX no solo para JAVA, sino para otros lenguajes de programación (Brownell, 2008).

En el *parser* que implemente funciones SAX, el usuario define cuando utilizar los diferentes métodos que son llamados cuando un evento ocurre mientras se va analizando el documento. Los eventos SAX incluyen (Means, y otros, 2008):

[4] **API** (**A**pplication **P**rogramming **I**nterfce) – Interfaz de Programación de Aplicaciones.
[5] **IDE** (**I**nterface **D**evelopment **E**nvironment) – Entorno de Desarrollo Integrado.
[6] ***Parser*** – Analizador gramatical.

- Nodos de texto XML
- Nodos de elementos XML
- Instrucciones de procesamiento XML
- Comentarios XML

Los eventos son lanzados cuando cada una de estas características XML son encontradas y nuevamente cuando se encuentra el final de éstas. El *parseo* se ejecuta de forma unidireccional, los datos previamente *parseados* no se pueden releer sin comenzar nuevamente la operación.

Como ventaja sobre DOM, SAX no almacena completamente el documento en memoria, lo que hace que sea más rápido y que utilice menos memoria (Brownell, 2008).

1.6.2. DOCUMENT OBJECT MODEL (DOM).

La traducción al español es Modelo de Objetos del Documento y es una convención para representar e interactuar con objetos en documentos HTML, XHTML y XML. Es multiplataforma e independiente del lenguaje (Hégaret, 2002).

Su estandarización fue llevada a cabo por la W3C justo después de ser creado **ECMAScript**, un estándar para lenguajes script en navegadores en 1997.

El primer estándar DOM reconocido fue el "DOM Level 1". Esta primera versión proveía de un modelo completo para documentos HTML o XML, incluyendo el cambiar cualquier porción de éste.

"DOM Level 2" fue publicado en el 2000; que introdujo soporte para **CSS** y en el 2004 salió la última versión conocida "DOM Level 3" que entre sus mejoras, cabe destacar, se incluye el manejo de eventos del teclado y una interface para la serialización[7] de documentos como XML.

Al igual que en SAX, los eventos que utiliza DOM son:

- Nodos de texto XML
- Nodos de elementos XML

[7]**Serialización** – Proceso de convertir una estructura de datos u objeto en una secuencia de bits para que puedan ser almacenadas en un archivo o en memoria, o transmitida por la red.

- Instrucciones de procesamiento XML
- Comentarios XML

DOM debe ser utilizado en aplicaciones donde se acceda al documento repetidamente o fuera de un orden secuencial, pues permite navegación en cualquier dirección y modificaciones arbitrarias. Por ello, es necesario que el árbol de datos del documento se almacene completamente en memoria.

1.7. ENTORNOS DE DESARROLLO INTEGRADO.

Un entorno de desarrollo integrado o IDE es un programa informático compuesto por un conjunto de herramientas de programación y puede utilizar varios lenguajes para ello.

Está compuesto básicamente por un editor de código, un compilador, un depurador y un constructor de interface gráfica.

Los IDE proveen un marco de trabajo para la mayoría de los lenguajes de programación. En algunos de estos los IDE pueden funcionar como un sistema en tiempo de ejecución, lo que permite utilizar el código de forma interactiva.

Dos de los IDE más utilizados actualmente son **Microsoft Visual Studio** y **Qt Creator**.

1.8. METODOLOGÍAS DE DESARROLLO DE SOFTWARE.

Las metodologías de desarrollo de software son usadas para planificar, estructurar y controlar el proceso de desarrollo del software. Existen varios modelos, paradigmas o filosofías de desarrollo en los cuales se puede apoyar para la realización de software, esto no quiere decir que una metodología se pueda utilizar para todos los proyectos de desarrollo.

Las metodologías están compuestas por:

- Una filosofía de desarrollo de software.
- Diversas herramientas, modelos y métodos.

A continuación se explican dos de las más utilizadas en la actualidad, Programación Extrema (**XP** – e**X**treme **P**rogramming) y Proceso Unificado de Rational[8] (**RUP – R**ational **U**nified **P**rocess):

1.8.1. XP.

Es una metodología de desarrollo de software que está dirigido a mejorar la calidad del software. XP prepara a los desarrolladores para responder con seguridad a las necesidades cambiantes de los clientes, incluso a finales del ciclo de vida (Wells, 2009).

Es una disciplina de desarrollo del software basada en la simplicidad, la comunicación y la retroalimentación. Su acción consiste en llevar al equipo de desarrollo con la presencia de prácticas sencillas, con información suficiente para que pueda observar donde se encuentran y ajustar las practicas a su situación personal (Ron, 2001).

1.8.2. RUP.

Es un proceso de desarrollo de software que junto al lenguaje UML, constituye la metodología más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Los principios de RUP son:

- Adaptar el proceso.
- Equilibrar las prioridades de los interesados.
- Colaboración entre equipos.
- Elevar el nivel de abstracción.
- Demostrar valor de forma iterativa.
- Centrarse constantemente en la calidad.

RUP no es un sistema de pasos, sino un conjunto de metodologías adaptables al contexto y necesidades de los clientes.

[8] **Rational** –Rational Software es un conjunto de software de IBM para el despliegue, diseño construcción, pruebas y administración de proyectos en el proceso de desarrollo de software. Inicialmente la compañía creadora de este paquete de software fue Rational Machines, fundada en 1981. Fue comprada por IBM en 2003.

El ciclo de vida RUP determina el ciclo de vida del proyecto que consiste en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto, además de hacer más o menos hincapié en cada actividad. En la **Figura 5** se muestra como varia el esfuerzo asociado las disciplinas según la fase en que se encuentre el proyecto.

Fases de RUP:

- **Inicio** o **Fase Inicial**: Determinar la visión del proyecto y a la validación inicial de este.
- **Elaboración**: Determinar la arquitectura óptima, mitigando los riesgos principales.
- **Construcción**: Construcción del software.
- **Transición**: Llevar el desarrollo, lo construido anteriormente, a la producción.

Este ciclo de vida se caracteriza por ser:

- **Dirigido por casos de uso**: Los casos de uso reflejan lo que los usuarios necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.
- **Centrado en la arquitectura**: La arquitectura muestra la visión del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción.
- **Iterativo e incremental**: Se propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros (Krebs, 2007).

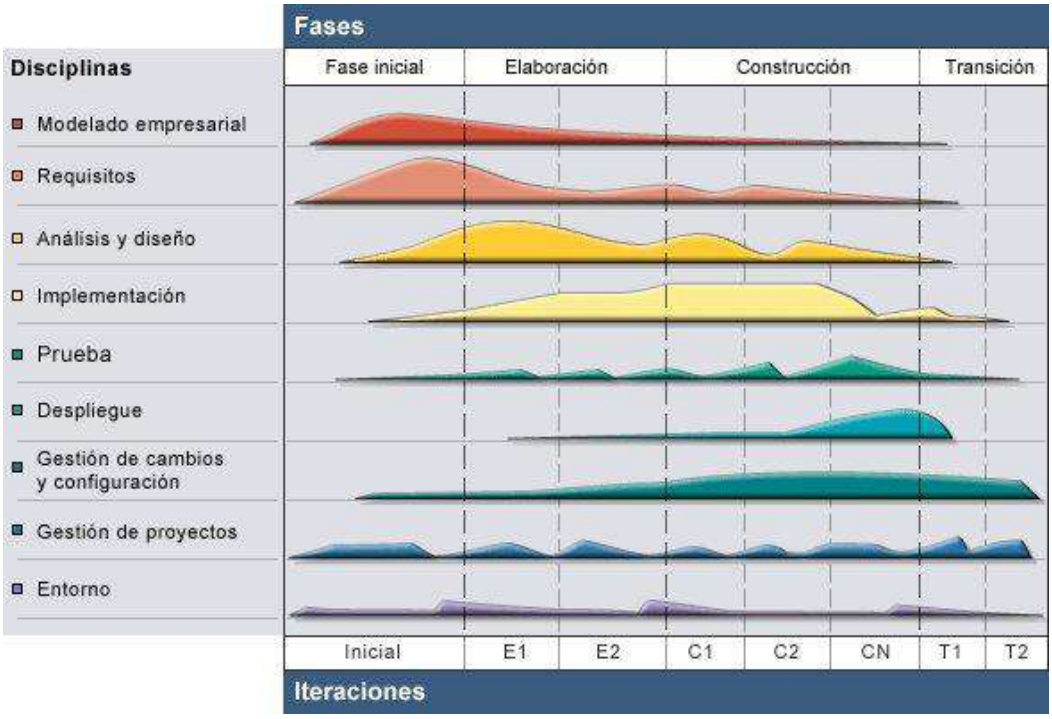

Figura 5: Arquitectura global de RUP.

CONCLUSIONES.

Este capítulo es un compendio de los principales conceptos y caracterizaciones que permiten una mejor comprensión del estado del arte de los temas principales a abordar en este trabajo. Ha permitido una primera aproximación al tema de los diálogos en los videojuegos y su tipología, así como el análisis de las interfaces de programación utilizadas en las formas de tratamiento de archivos XML y de las metodologías de desarrollo de software.

CAPÍTULO 2: SOLUCIÓN PROPUESTA.

INTRODUCCIÓN.

En este capítulo se describe la propuesta para dar solución a la necesidad actual de un sistema de diálogos para los Entrenadores Aduaneros. Aquí se exponen las herramientas, el lenguaje y la metodología a seguir para la creación del sistema.

2.1. PROPUESTA DEL SISTEMA.

El sistema de diálogo a desarrollar, estará conformado por dos módulos:

- El primero, el **EDEditor** (**E**asy **D**ialog **Editor**) será utilizado para la gestión de los archivos contenedores de los diálogos, en este caso, archivos con extensión XML.
- El segundo será el cargador que utilizará **Entrenadores Aduaneros** para gestionar los diálogos dentro del sistema y se denominará **EDCore** (**E**asy **D**alog **Core**).

Todo sistema de diálogos no necesita un editor personalizado para éste, pero en este caso la forma en que se organizarán los textos y los atributos, necesita de una herramienta que respondiese a sus intereses.

La **Figura 6** representa esquemáticamente cómo interaccionarán estos módulos con el sistema **Entrenadores Aduaneros**.

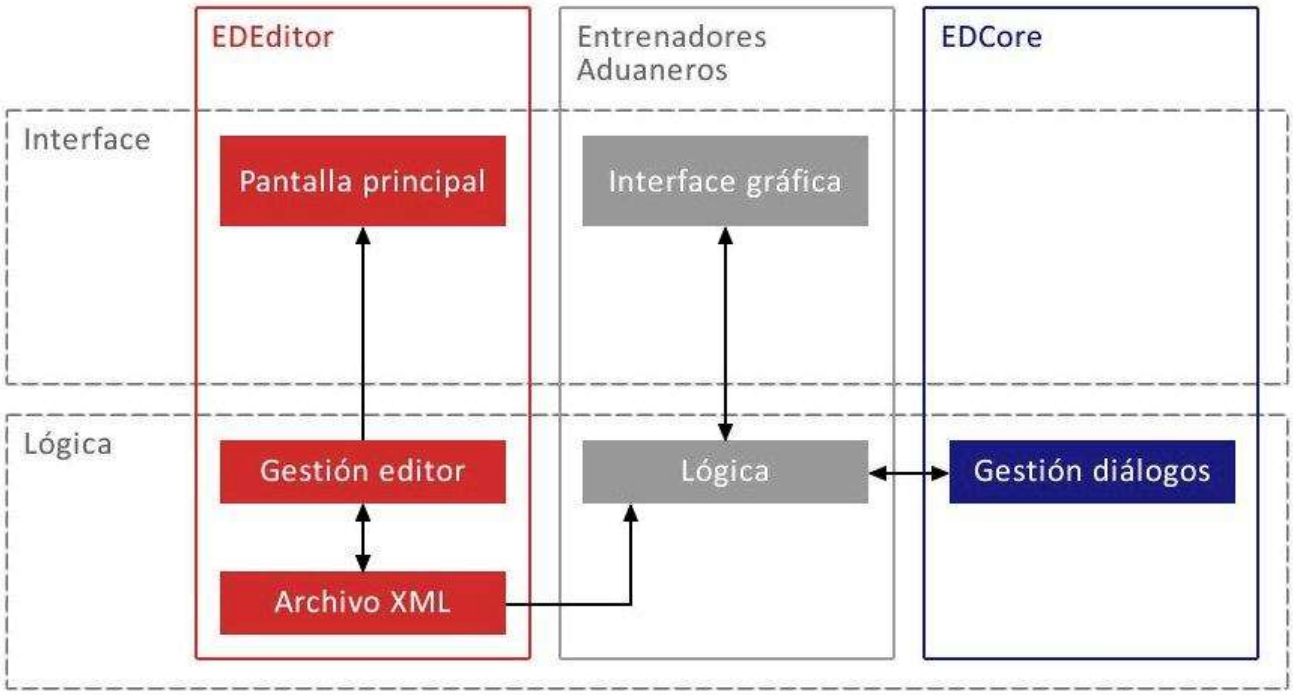

Figura 6: Vista lógica: Sistema-Módulos.

2.2. CLASIFICACIONES DE DIÁLOGO Y FUNCIÓN DE CONTROL A UTILIZAR.

Como se pudo apreciar en el capítulo anterior, varias son las formas de clasificar los **sistemas de diálogo** y además de **funciones de control** para éstos.

Después de un estudio sobre la forma de intercambio de información dentro del sistema **Entrenadores Aduaneros**, se llegó a la conclusión de que las clasificaciones de diálogo que caracterizarán al nuevo sistema en creación son: **basado en estados finitos** y **basado en formas**. Al fusionarlos y simplificar algunas de sus características, se logrará facilidad de uso y adaptación.

Se utilizará además la función de control de diálogos: **diálogo de iniciativa mixta**. Sus principales características y el porqué de su uso, se explica con mayor extensión a continuación.

2.2.1. CLASIFICACIÓN DE DIÁLOGO: BASADO EN ESTADOS FINITOS.

Como se puntualizó anteriormente, esta clasificación se aplica cuando el sistema de diálogo puede ser modelado como un grafo formado por nodos representando estados y con arcos entre éstos que denotan transiciones. Pero en el caso de los **Entrenadores Aduaneros**, el usuario no es el que pasará de un estado a otro cuando se cumpla la transición correspondiente, sino el **NPC**. En la situación siguiente se ve con más claridad.

En caso de que un **PC** se dirija a un **NPC** para pedirle ver que trae en el equipaje de mano y este contenga por ejemplo, explosivos, las variables que cambiarán serán las mostradas en la **Figura 7**, el **nivel de pánico** (**panic**), de **Normal** a **Alto** y la **expresión facial** (**face**), de **Tranquilo** a **Incómodo**.

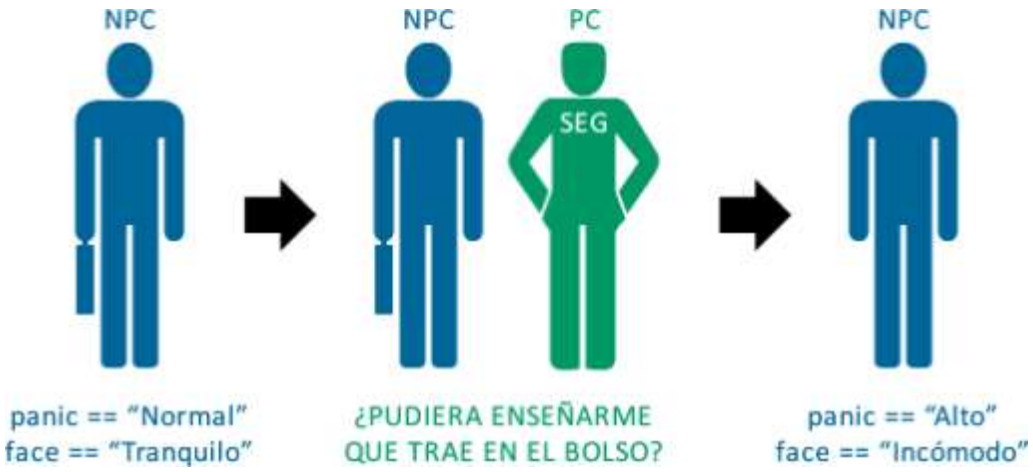

Figura 7: Ejemplo de tipo de diálogo: basado en estados finitos.

Como se puede apreciar, se utiliza el término “variables” en vez de “estados” cuando se refiere al trabajo con los diálogos. Esto se debe a que el módulo de gestión de diálogos, EDCore, no trabajará basándose en estados porque el sistema no requiere de mayor complejidad a la hora de acceder y mostrar la información.

Al utilizar esta variación, la cantidad de iteraciones para acceder a la información se reduce significativamente. Prueba de ello son las figuras que se muestran a continuación. La **Figura 8** ilustra cómo quedaría la estructura del archivo XML en caso de utilizar estados

finitos y la **Figura 9** como quedaría utilizando la variante propuesta para **Entrenadores Aduaneros**, mostrando así una buena optimización.

```
<Dialogue>
    <String ID="1" Type="Question" Text="Preg1" NextID="2,3,0">
        <Vars PanicLevel="Normal" Face="Non" NextIDRandom="true"/>
    </String>
    <String ID="2" Type="Response" Text="Resp1.1" NextID="Begin">
        <Vars PanicLevel="Normal" Face="Smile" NextIDRandom="false"/>
    </String>
    <String ID="3" Type="Response" Text="Resp1.2" NextID="Begin">
        <Vars PanicLevel="Normal" Face="Serious" NextIDRandom="false"/>
    </String>
    <String ID="4" Type="Question" Text="Preg2" NextID="5,6,0">
        <Vars PanicLevel="Normal" Face="Non" NextIDRandom="true"/>
    </String>
    <String ID="5" Type="Response" Text="Resp2.1" NextID="Begin">
        <Vars PanicLevel="Normal" Face="Sad" NextIDRandom="false"/>
    </String>
    <String ID="6" Type="Response" Text="Resp2.2" NextID="Begin">
        <Vars PanicLevel="Normal" Face="Smile" NextIDRandom="false"/>
    </String>
    <String ID="0" Type="Close" Text="Terminar" NextID="Non">
        <Vars PanicLevel="Non" Face="Non" NextIDRandom="false"/>
    </String>
</Dialogue>
```

Figura 8: Ejemplo de almacenamiento de un diálogo en un archivo XML usando estados finitos.

```
<Dialogue>
    <PanicLevel Level="Normal">
        <Question ID="1" Text=" Preg1">
            <Answer ID="1" Text="Resp1.1" Face="Smile" />
            <Answer ID="2" Text="Resp1.2" Face="Serious" />
        </Question>
        <Question ID="2" Text=" Preg2>">
            <Answer ID="1" Text="Resp2.1" Face="Sad" />
            <Answer ID="2" Text="Resp2.2" Face="Smile" />
        </Question>
    </PanicLevel>
</Dialogue>
```

Figura 9: Ejemplo de almacenamiento de un diálogo en un archivo XML usando un método creado específicamente para Entrenadores Aduaneros.

De esta manera se cumplen los factores de los cuales depende la selección de la tecnología para un gestor de diálogos.

2.2.2. CLASIFICACIÓN DE DIÁLOGO: BASADO EN FORMAS.

Como anteriormente se expresó, en este tipo de diálogo se pueden modelar, para una misma entrada, varios estados, así como manipular las entradas en diferente ordenamiento. En el caso de los **Entrenadores Aduaneros** el **NPC**, de forma aleatoria o no y dependiendo de la configuración que se le dé, responderá a una situación específica. No todos pensamos de la misma manera ni siempre tomamos la opción correcta y esta premisa, llevada al sistema en desarrollo, logra veracidad. La situación que se ve a continuación explica cómo se usa este tipo de diálogo.

Al simular la salida de un **NPC** de la zona de recogida de equipaje, se puede ejecutar la acción de que pida orientación a un **PC** para ir al baño más cercano o a una cabina telefónica o, sin pedir indicación alguna, realizar las acciones anteriores.

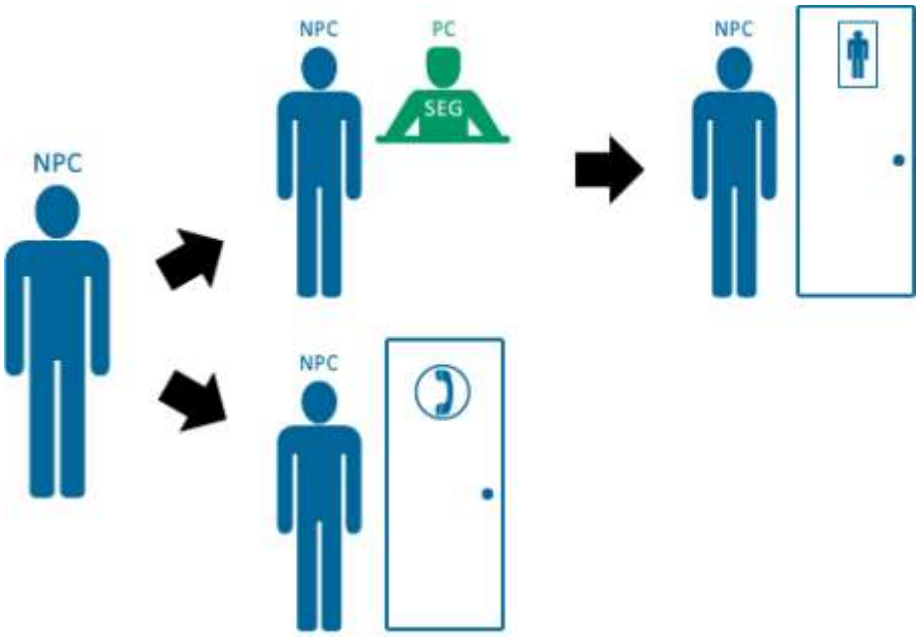

Figura 10: Ejemplo de tipo de diálogo: basado en formas.

2.2.3. FUNCIÓN DE CONTROL: DIÁLOGO DE INICIATIVA MIXTA.

Un diálogo es de iniciativa mixta cuando cualquiera de los agentes presentes en un entorno, **NPC** o **PC**, puede iniciar la conversación.

Esta función será la escogida como rectora de las interacciones en el sistema pues se pueden gestionar diversas situaciones en éste que responden perfectamente a la necesidad de **Entrenadores Aduaneros** de que tanto los **PC** como los **NPC** puedan iniciar una conversación. Prueba de ello son las dos situaciones vistas anteriormente en la **Figura 7** y la **Figura 10**. En la primera el **PC** es el que inicia la conversación con el **NPC** interrumpiendo su trayectoria y en el segundo el **NPC** va a pedir ayuda a un **PC**.

2.3. INTERFACE DE PROGRAMACIÓN, HERRAMIENTA Y METODOLOGÍA DE DESARROLLO A UTILIZAR.

La selección de QtCreator como IDE para el desarrollo de los módulos, viene dada por su calidad de contener un módulo para el trabajo con archivos XML, específicamente con la interface de programación DOM que responde a las necesidades del sistema, por la ampliación de las funcionalidades que le ha propiciado Qt al lenguaje C++ y por ser libre.

RUP es la metodología de desarrollo seleccionada por ser iterativo e incremental, centrado en arquitectura y dirigido por casos de uso; además por presentar interés fundamental en el análisis y diseño.

CONCLUSIONES.

En este capítulo se propuso la forma en que interactuarán los módulos con el **Sistema de Entrenadores Aduaneros**. Se seleccionaron los tipos de diálogo: **basado en estados finitos** y **basado en formas** y la función de control: **diálogo de iniciativa mixta**, que son las que más se ajustan a la necesidad de la aplicación. Aunque para lograr una mayor eficiencia y adaptarse a la simplicidad actual del **Sistema de Entrenadores Aduaneros**, en el caso del tipo de diálogo: **basado en estados finitos**, se realizó una modificación que permitirá no solo lo puntualizado anteriormente, sino también menor tiempo de acceso a la información de los ficheros de diálogo.

Además, se seleccionaron la herramienta y la metodología de desarrollo a utilizar en el proceso de confección de los módulos.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA.

INTRODUCCIÓN.

En el presente capítulo se presenta el modelo de dominio, los actores que interactuarán con el sistema, los requisitos funcionales y no funcionales que debe tener éste y se definen los casos de uso. Además se presentan los diagramas de casos de uso para representar la interacción de los actores con el sistema y se realiza una descripción textual de estas interacciones. Además se presentan los diagramas de clases del diseño y los diagramas de componentes.

3.1. MODELO DE DOMINIO.

El modelo de dominio se describe mediante diagramas UML y es una representación visual de los conceptos u objetos significativos para un problema o área de interés.

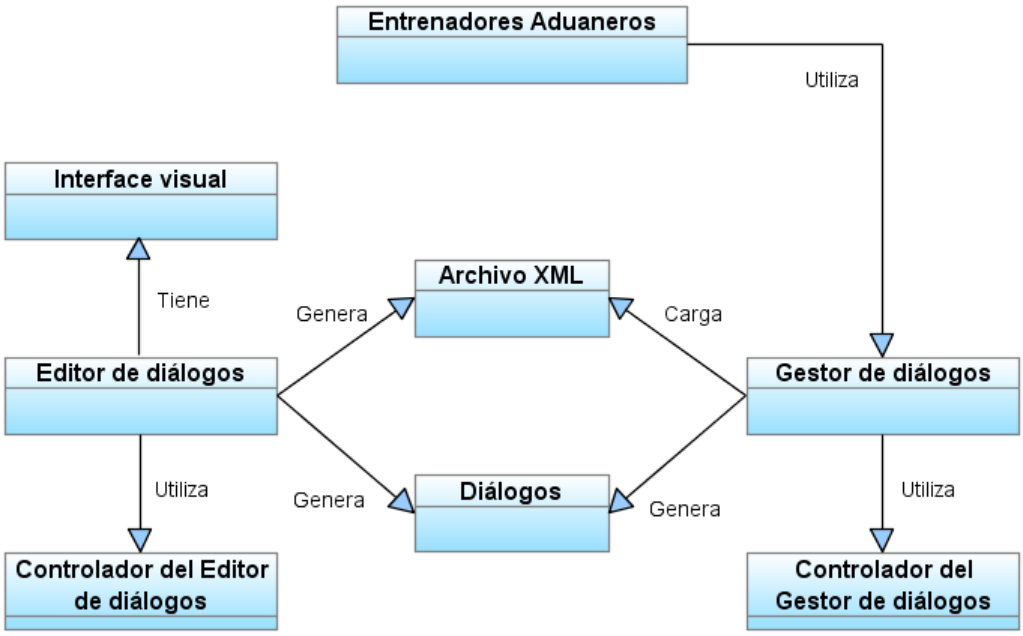

Figura 11: Modelo de dominio.

3.2. GLOSARIO DE TÉRMINOS DEL DOMINIO.

A continuación se exponen los términos utilizados en el modelo de dominio.

- **Entrenadores Aduaneros**: Proyecto al que tributa el sistema de diálogos.
- **Editor de diálogos**: Módulo para la creación de los diálogos.
- **Interface visual**: Interface del programa editor de diálogos.
- **Controlador del Editor de diálogos**: Gestor del módulo editor de diálogos.
- **Gestor de diálogos**: Módulo de gestión de diálogos.
- **Controlador del Gestor de diálogos**: Gestor del módulo Gestor de diálogos.
- **Archivo XML**: Archivo que genera el módulo **Editor de diálogos** y que utiliza el módulo **Gestor de diálogos**.
- **Diálogo**: Información con la cual trabajan los módulos. Se almacena en **Archivo XML**.

3.3. CAPTURA DE REQUISITOS.

A continuación se exponen los requisitos funcionales y no funcionales del sistema.

3.3.1. REQUISITOS FUNCIONALES.

Al existir dos módulos y cada uno tener requisitos funcionales propios y comunes entre ambos, se listarán en dos grupos.

Requisitos funcionales del módulo **Easy Dialog Editor**:

- **R1** – Gestionar pregunta.
  - **R1.1** – Crear pregunta.
  - **R1.2** – Modificar pregunta.
  - **R1.3** – Eliminar pregunta.
- **R2** – Gestionar respuesta.
  - **R2.1** – Crear respuesta.
  - **R2.2** – Modificar respuesta.
  - **R2.3** – Eliminar respuesta.
- **R3** – Gestionar archivo XML.
  - **R3.1** – Nuevo archivo XML.

- **R3.2** – Guardar archivo XML.

Requisitos funcionales del módulo **Easy Dialog Core**:

- **R4** – Obtener pregunta.
- **R5** – Obtener respuesta.
- **R6** – Obtener expresión.
- **R7** – Cargar archivo XML.

3.3.2. REQUISITOS NO FUNCIONALES.

Para la herramienta a desarrollar, se especifican los siguientes requisitos no funcionales:

- **Usabilidad**: En caso del módulo **Easy Dialog Editor**, el lingüista o editor que trabaje con la herramienta, solo necesita conocimientos básicos informáticos. Por otra parte, los programadores que trabajen con el módulo **Easy Dialog Core**, deben tener conocimientos básicos de programación.
- **Rendimiento**: Ambos módulos deben ser ágiles en manejar los datos que gestionan.
- **Soporte:** En el caso específico del módulo **Easy Dialog Core**, debe ser compatible con las plataformas Windows y Linux. Con respecto al **Easy Dialog Editor**, próximamente se podría, con algunas modificaciones, permitir su utilización en Linux.
- **Hardware:** Al ser módulos que utilizan muy pocos recursos de procesamiento, no se necesitará una configuración de hardware específica.
- **Diseño e implementación:** Se regirá por la filosofía de **Programación Orientada a Objetos** utilizando el framework **Qt**.
- **Extensibilidad:** Se regirá por la necesidad del sistema al cual tributa.

3.4. DEFINICIÓN DE LOS CASOS DE USO.

Habiendo recopilado los requisitos, se exponen a continuación los actores del sistema y los casos de uso que facilitan una descripción de cómo se utilizará el sistema.

3.4.1. ACTORES DEL SISTEMA.

Los actores que van a interactuar con los módulos se muestran a continuación.

Actores	Justificación
Lingüista o Editor	Es el que hará uso del programa **EDEditor**, creando y modificando los archivos que almacenan los diálogos.
Programador	Es la que hará uso del sistema, recuperando los datos almacenados en los archivos de diálogo mediante la utilización del módulo **EDCore**.

Tabla 1: Actores del sistema.

3.4.2. CASOS DE USO DEL SISTEMA.

Casos de uso del módulo **Easy Dialog Editor**:

- **CU1** – Gestionar pregunta.
- **CU2** – Gestionar respuesta.
- **CU3** – Gestionar archivo XML.

Casos de uso del módulo **Easy Dialog Core**:

- **CU4** – Obtener pregunta.
- **CU5** – Obtener respuesta.
- **CU6** – Obtener expresión.
- **CU7** – Cargar archivo XML.

3.4.3. DIAGRAMAS DE CASOS DE USO DEL SISTEMA.

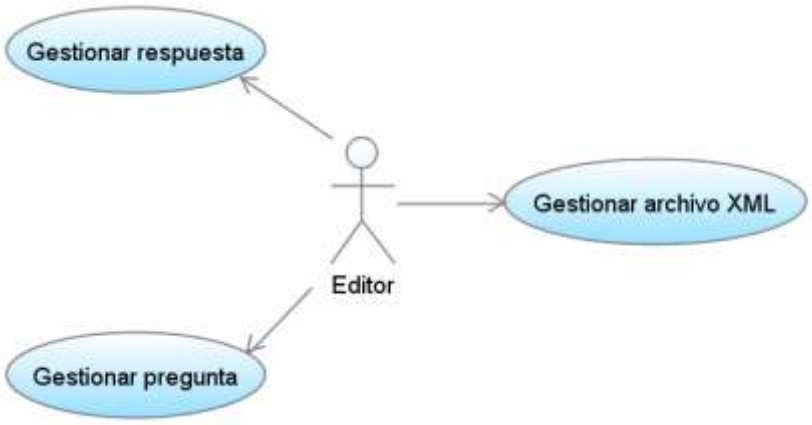

Figura 12: Diagrama de casos de uso del módulo Easy Dialog Editor.

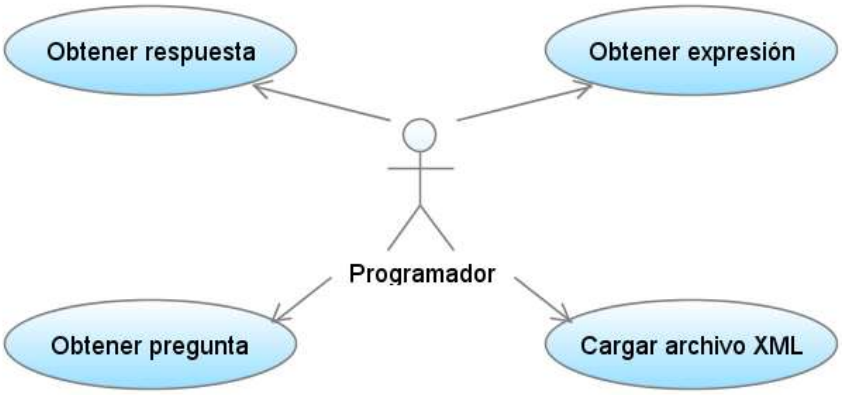

Figura 13: Diagrama de casos de uso del módulo Easy Dialog Core.

3.4.4. DESCRIPCIÓN TEXTUAL DE LOS CASOS DE USO.

En esta sección se describen la secuencia de eventos que los actores realizan para completar un proceso. Se exponen tanto las acciones que realizan los actores sobre el sistema, como las respuestas que surgen de este como consecuencia de cada evento.

Casos de uso expandidos para el módulo Easy Dialog Editor.

Caso de uso	
CU-1	Gestionar pregunta.
Propósito	El objetivo del caso de uso es que el editor o lingüista pueda gestionar el pregunta.
Prioridad	Crítica.
Actores:	Editor.
Resumen:	El actor inicializa el caso de uso cuando selecciona una de las siguientes acciones: Crear pregunta, Modificar pregunta o Eliminar pregunta.
Referencias	R1, R1.1, R1.2, R1.3
Precondiciones	
Poscondiciones	Se crea, modifica o elimina una pregunta.
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El caso de uso se inicializa cuando se	1.1. Si el actor selecciona la opción

selecciona una de las siguientes acciones: Crear pregunta, Modificar pregunta o Eliminar pregunta.	Crear pregunta, ir a la Acción 1 Crear pregunta. 1.2. Si el actor selecciona la opción Modificar pregunta, ir a la Acción 2 Modificar pregunta. 1.3. Si el actor selecciona la opción Eliminar pregunta, ir a la Acción 3 Eliminar pregunta.
Acción 1 Crear pregunta	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando el actor selecciona el nivel de pánico que va a contener la pregunta. 2. El actor teclea el texto de la pregunta. 3. El actor da clic en la opción Crear pregunta.	1.1. El sistema habilita el campo donde se insertará el texto de la pregunta. 3.1. El sistema crea una nueva pregunta en el nivel de pánico seleccionado. Finaliza el caso de uso.
Curso alterno de eventos	
Acción del autor	**Respuesta del sistema**
2. El actor no marca un nivel de pánico. 3. El actor escribe un texto de pregunta que ya existe en este nivel de pánico.	2.1. El sistema no habilita el campo para insertar el texto. 3.1. El sistema muestra un mensaje indicando que ya existe esta pregunta en el nivel de pánico seleccionado. Finaliza el caso de uso.
Acción 2 Modificar pregunta	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando el actor selecciona la pregunta que desea modificar.	1.1. El sistema habilita el campo donde se modificará el texto de la pregunta. 3.1. El sistema modifica el texto de la

2. El actor cambia el texto existente por el que desee. 3. El actor da clic en la opción Modificar pregunta.	pregunta seleccionada. Finaliza el caso de uso.
Curso alterno de eventos	
Acción del actor	**Respuesta del sistema**
4. El actor escribe un texto de pregunta que ya existe en este nivel de pánico.	4.1. El sistema muestra un mensaje indicando que ya existe esta pregunta en el nivel de pánico seleccionado. Finaliza el caso de uso.
Acción 3 Eliminar pregunta	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando el actor selecciona la pregunta que va a eliminar. 2. El actor da clic en eliminar pregunta.	2.1. El sistema elimina la pregunta. Finaliza el caso de uso.
Curso alterno de eventos	
Acción del actor	**Respuesta del sistema**
-	-

Tabla 2: Descripción textual CU-1.

Caso de uso	
CU-2	Gestionar respuesta.
Propósito	El objetivo del caso de uso es que el editor o lingüista pueda gestionar una respuesta.
Prioridad	Crítica.
Actores:	Editor.
Resumen:	El actor inicializa el caso de uso cuando selecciona una de las siguientes acciones: Crear respuesta, Modificar respuesta o Eliminar respuesta.

Referencias	R2, R2.1, R2.2, R2.3
Precondiciones	
Poscondiciones	Se crea, modifica o elimina una respuesta.

Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El caso de uso se inicializa cuando se selecciona una de las siguientes acciones: Crear respuesta, Modificar respuesta o Eliminar respuesta.	1.1. Si el actor selecciona la opción Crear respuesta, ir a la Acción 1 Crear respuesta. 1.2. Si el actor selecciona la opción Modificar respuesta, ir a la Acción 2 Modificar respuesta. 1.3. Si el actor selecciona la opción Eliminar respuesta, ir a la Acción 3 Eliminar respuesta.
Acción 1 Crear respuesta	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando el actor selecciona la pregunta que va a contener la respuesta. 2. El actor teclea el texto de la respuesta y el texto de la expresión correspondiente. 3. El actor da clic en la opción Crear respuesta.	1.1. El sistema habilita los campos donde se insertarán los textos de la respuesta. 3.1. El sistema crea una nueva respuesta en la pregunta seleccionada. Finaliza el caso de uso.
Curso alterno de eventos	
Acción del autor	**Respuesta del sistema**
4. El actor no marca una pregunta. 5. El actor escribe un texto de respuesta que ya existe en esta pregunta.	4.1. El sistema no habilita los campos para insertar los textos. 5.1. El sistema muestra un mensaje indicando que ya existe esta respuesta en la pregunta seleccionada. Finaliza el caso de

	uso.
Acción 2 Modificar respuesta	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando el actor selecciona la respuesta que desea modificar. 2. El actor cambia el texto existente en el campo de texto respuesta o expresión o en ambos por el o los que desee. 3. El actor da clic en la opción Modificar respuesta.	1.1. El sistema habilita los campos donde se modificarán los textos de la respuesta. 3.1. El sistema modifica el o los textos de la respuesta seleccionada. Finaliza el caso de uso.
Curso alterno de eventos	
Acción del actor	**Respuesta del sistema**
4. El actor escribe un texto de respuesta que ya existe en esta pregunta.	4.1. El sistema muestra un mensaje indicando que ya existe esta respuesta en la pregunta seleccionada. Finaliza el caso de uso.
Acción 3 Eliminar respuesta	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando el actor selecciona la respuesta que va a eliminar. 2. El actor da clic en eliminar respuesta.	2.1. El sistema elimina la respuesta. Finaliza el caso de uso.
Curso alterno de eventos	
Acción del actor	**Respuesta del sistema**
-	-

Tabla 3: Descripción textual CU-2.

Caso de uso	
CU-3	Gestionar archivo XML.
Propósito	El objetivo del caso de uso es que el editor o lingüista pueda gestionar un archivo XML.
Prioridad	Crítica.
Actores:	Editor.
Resumen:	El actor inicializa el caso de uso cuando selecciona una de las siguientes acciones: Nuevo diálogo o Guardar diálogo.
Referencias	R3, R3.1, R3.2.
Precondiciones	
Poscondiciones	Se crea o modifica un archivo XML.

Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El caso de uso se inicializa cuando se selecciona una de las siguientes acciones: Nuevo diálogo o Guardar diálogo.	1.1. Si el actor selecciona la opción Nuevo diálogo, ir a la Acción 1 Nuevo diálogo. 1.2. Si el actor selecciona la opción Guardar diálogo, ir a la Acción 2 Guardar diálogo.
Acción 1 Nuevo diálogo	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando el actor selecciona la opción Nuevo diálogo.	1.1. El sistema actualiza su interface visual.
Curso alterno de eventos	
Acción del autor	**Respuesta del sistema**
2. El actor selecciona la opción Nuevo diálogo cuando se encuentra abierto uno existente. 3. El actor selecciona la opción Guardar diálogo abierto. 4. El actor selecciona donde Guardar el	2.1. El sistema da la opción de Guardar diálogo abierto o cancelar la acción. 3.1. El sistema pregunta al usuario donde Guardar el diálogo abierto. 4.1. El sistema Guarda el diálogo abierto en la ubicación dada por el actor y

diálogo abierto.	actualiza la interface visual. Finaliza el caso de uso.
Acción 2 Guardar diálogo	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando el actor selecciona la opción Guardar diálogo. 2. El actor escribe el nombre del archivo que va a contener el diálogo y la ubicación donde lo desea guardar. 3. El actor da clic en la opción Guardar diálogo.	1.1. El sistema pregunta donde el usuario guardará el diálogo y con qué nombre. 3.1. El sistema guarda el diálogo donde eligió el actor. Finaliza el caso de uso.
Curso alterno de eventos	
Acción del actor	**Respuesta del sistema**
-	-

Tabla 4: Descripción textual CU-3.

Casos de uso expandidos para el módulo Easy Dialog Core.

Caso de uso	
CU-4	Obtener preguntas.
Propósito	El objetivo del caso de uso es que el programador pueda obtener la lista de preguntas pertenecientes a un nivel de pánico.
Prioridad	Crítica.
Actores:	Programador.
Resumen:	El actor inicializa el caso de uso cuando solicita la lista de preguntas de un nivel de pánico.
Referencias	R4.
Precondiciones	Crear objeto tipo EDCore y cargar archivo XML.
Poscondiciones	Se obtiene una pregunta.
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**

1. El caso de uso se inicializa cuando se solicita la lista de preguntas.	1.1. Si el actor selecciona la opción Obtener preguntas, ir a la Acción 1 Obtener preguntas.
Acción 1 Obtener preguntas	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando se solicita la lista de preguntas de un nivel de pánico específico.	1.1. El sistema devuelve la lista de preguntas correspondiente.
Curso alterno de eventos	
Acción del autor	**Respuesta del sistema**
	2.1. El sistema devuelve la lista vacía.

Tabla 5: Descripción textual CU-4.

Caso de uso	
CU-5	Obtener respuestas.
Proposito	El objetivo del caso de uso es que el programador pueda obtener la lista de respuestas pertenecientes a una pregunta.
Prioridad	Crítica.
Actores:	Programador.
Resumen:	El actor inicializa el caso de uso cuando solicita la lista de respuestas de una pregunta.
Referencias	R5.
Precondiciones	Crear objeto tipo EDCore y cargar archivo XML.
Poscondiciones	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El caso de uso se inicializa cuando se solicita la lista de respuestas.	1.1. Si el actor selecciona la opción Obtener respuestas, ir a la Acción 1 Obtener respuestas.
Acción 1 Obtener respuestas	

Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando se solicita la lista de respuestas de una pregunta y nivel de pánico específicos.	1.1. El sistema devuelve la lista de respuestas correspondiente.
Curso alterno de eventos	
Acción del autor	**Respuesta del sistema**
	2.1. El sistema devuelve la lista vacía.

Tabla 6: Descripción textual CU-5.

Caso de uso	
CU-6	Obtener expresión.
Proposito	El objetivo del caso de uso es que el programador pueda obtener la expresión perteneciente a una respuesta.
Prioridad	Crítica.
Actores:	Programador.
Resumen:	El actor inicializa el caso de uso cuando solicita la expresión de una respuesta.
Referencias	R6.
Precondiciones	Crear objeto tipo EDCore y cargar archivo XML.
Poscondiciones	Se obtiene una respuesta.
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El caso de uso se inicializa cuando se solicita la expresión de una respuesta.	1.1. Si el actor selecciona la opción Obtener expresión, ir a la Acción 1 Obtener expresión.
Acción 1 Obtener expresión	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando se solicita la expresión de una pregunta , respuesta	1.1. El sistema devuelve la expresión de la respuesta correspondiente.

y nivel de pánico específicos.	
Curso alterno de eventos	
Acción del autor	**Respuesta del sistema**
	2.1. El sistema no devuelve nada.

Tabla 7: Descripción textual CU-6.

Caso de uso	
CU-7	Cargar archivo XML.
Propósito	El objetivo del caso de uso es que el programador cargue el archivo XML.
Prioridad	Crítica.
Actores:	Editor.
Resumen:	El actor inicializa el caso de uso cuando se ejecuta el método Cargar archivo XML.
Referencias	R7.
Precondiciones	Crear objeto tipo EDCore.
Poscondiciones	Se carga un archivo XML.
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El caso de uso se inicializa cuando se pasa por parámetro la dirección del archivo XML.	1.1. Si el actor selecciona la opción Cargar archivo XML, ir a la Acción 1 Cargar archivo XML.
Acción 1 Nuevo diálogo	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. La acción se inicializa cuando se pasa por parámetro al método Cargar archivo XML la dirección de donde se encuentra el archivo en cuestión.	1.2. El sistema almacena la información del archivo en un objeto.
Curso alterno de eventos	
Acción del autor	**Respuesta del sistema**

2. La dirección pasada por parámetro es incorrecta o no existe.	2.1. El sistema no almacena información alguna en el objeto.

Tabla 8: Descripción textual CU-7.

3.5. DIAGRAMA DE CLASES DE DISEÑO.

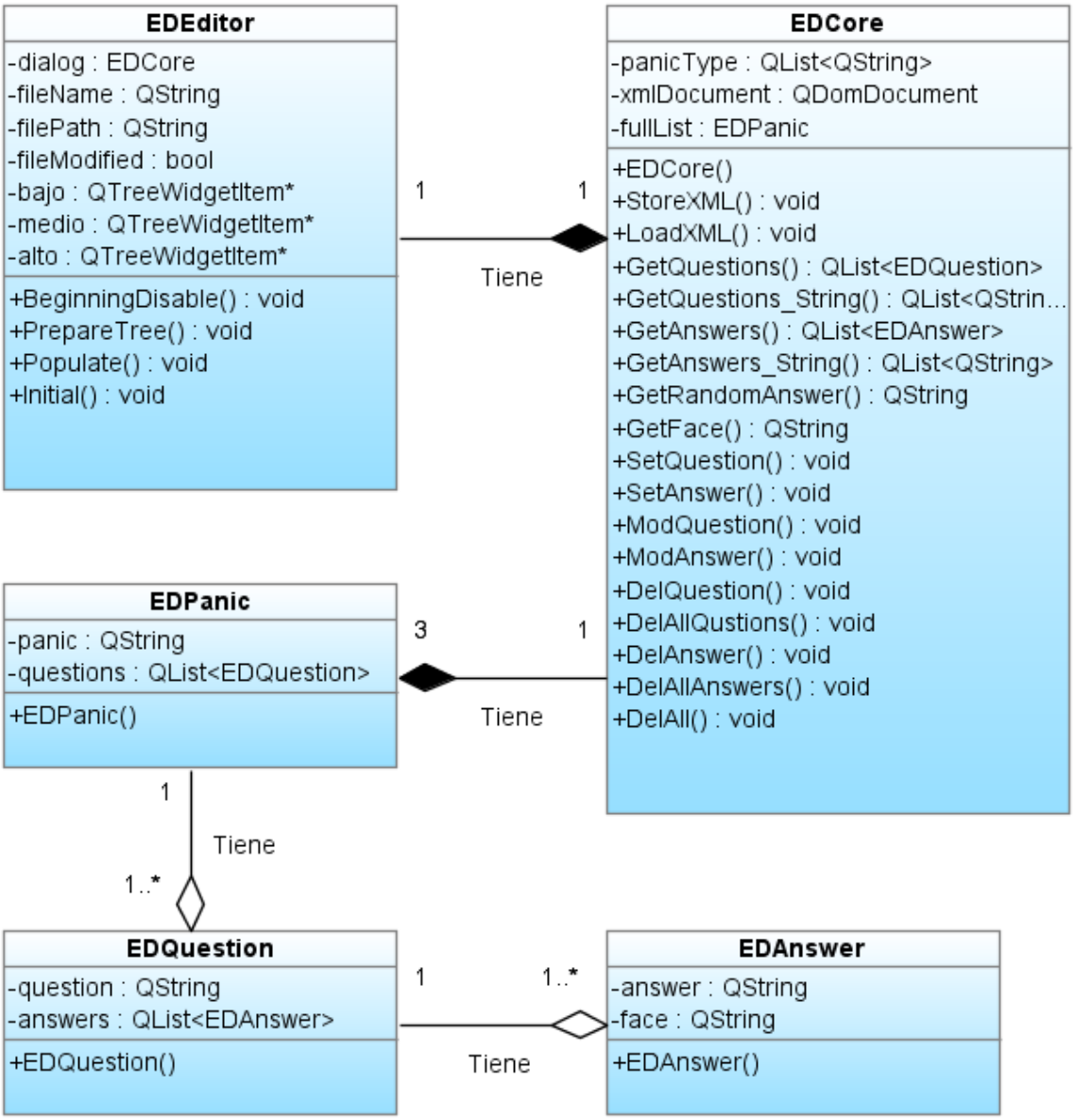

Figura 14: Diagrama de clases de Diseño.

3.6. DIAGRAMA DE COMPONENTES.

Los diagramas de componentes modelan la vista estática del sistema. En ellos se muestra la organización y las dependencias lógicas entre el conjunto de componentes de software, ya sean éstos código fuente, bibliotecas o ejecutables. Para su elaboración se tienen en cuenta los requisitos relacionados con la facilidad de desarrollo, la gestión de software, reutilización, el lenguaje de programación y las herramientas utilizadas.

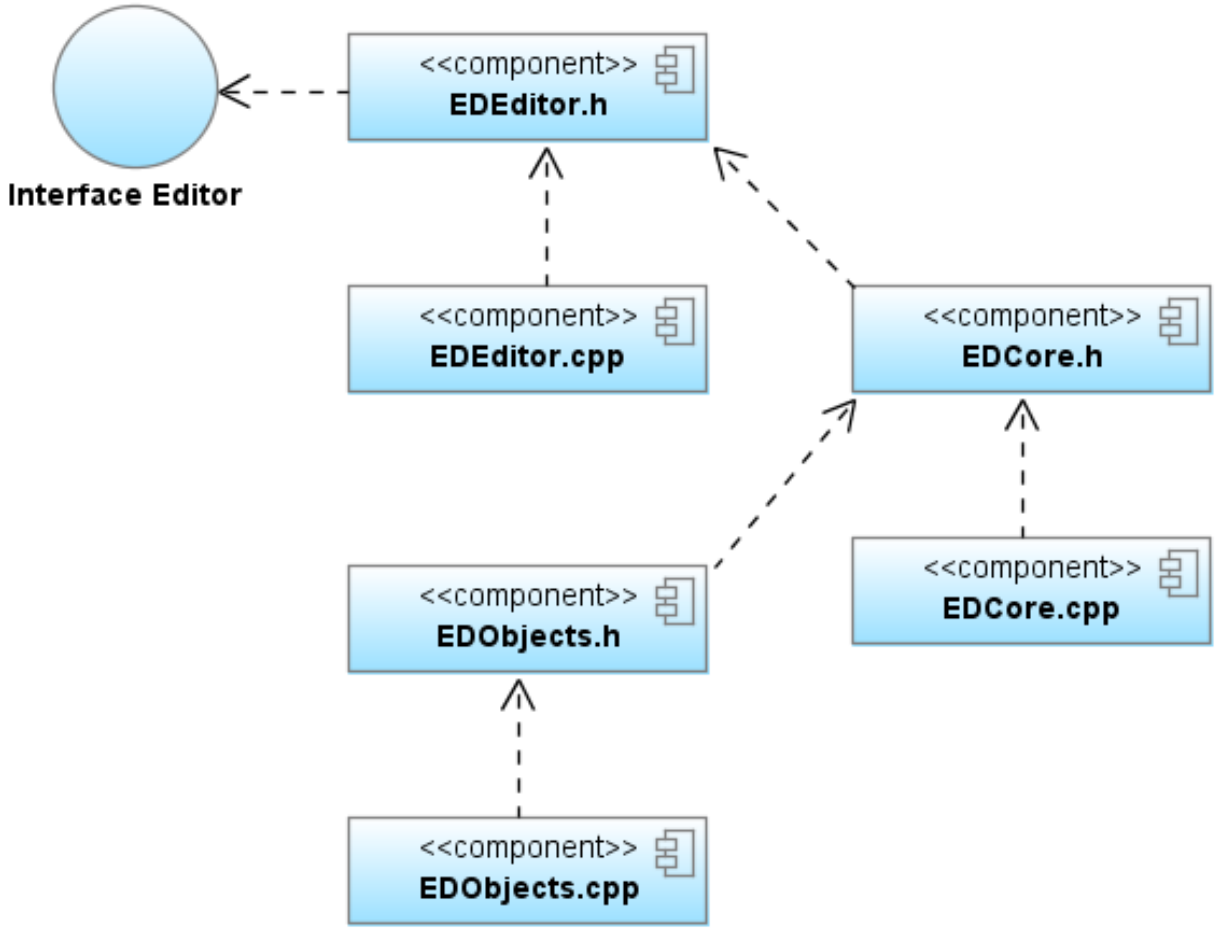

Figura 15: Diagrama de componentes. Módulo EDEditor.

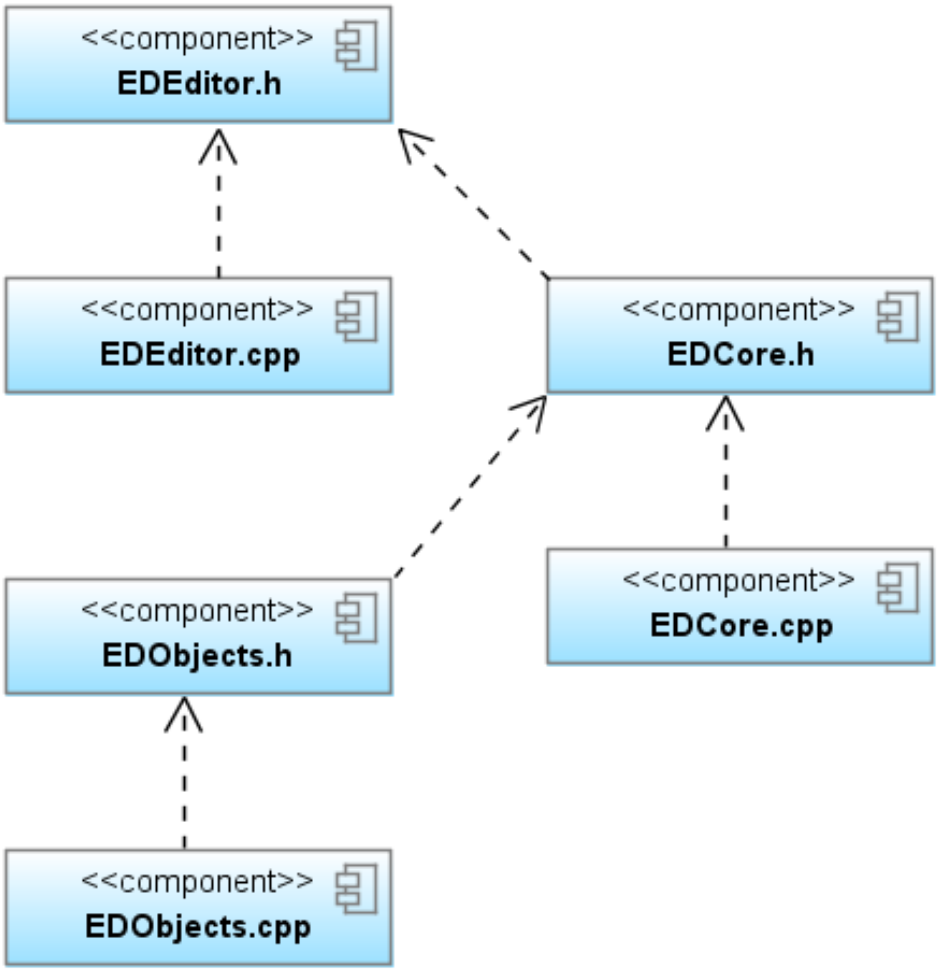

Figura 16: Diagrama de componentes. Módulo EDCore.

CONCLUSIONES.

En este capítulo se presentó el modelo de domino para obtener una mejor comprensión de los conceptos y procesos que se manejan dentro del negocio. Se describieron los requisitos que la herramienta debe cumplir para su funcionamiento óptimo y se presentaron los casos de uso del sistema y la descripción textual de cada uno de ellos con el objetivo de especificar las funcionalidades del sistema.

De forma general, se logró describir el sistema que se desea desarrollar, lo que deja las bases sentadas para la etapa de desarrollo.

ANEXOS.

PROPUESTA DEL MÓDULO EDEDITOR.

Las propuestas de interface que se verán a continuación utilizan diversas características como simplicidad y colores representativos para lograr la mejor adaptación de los editores o lingüistas que puedan trabajar con el proyecto **Entrenadores Aduaneros**.

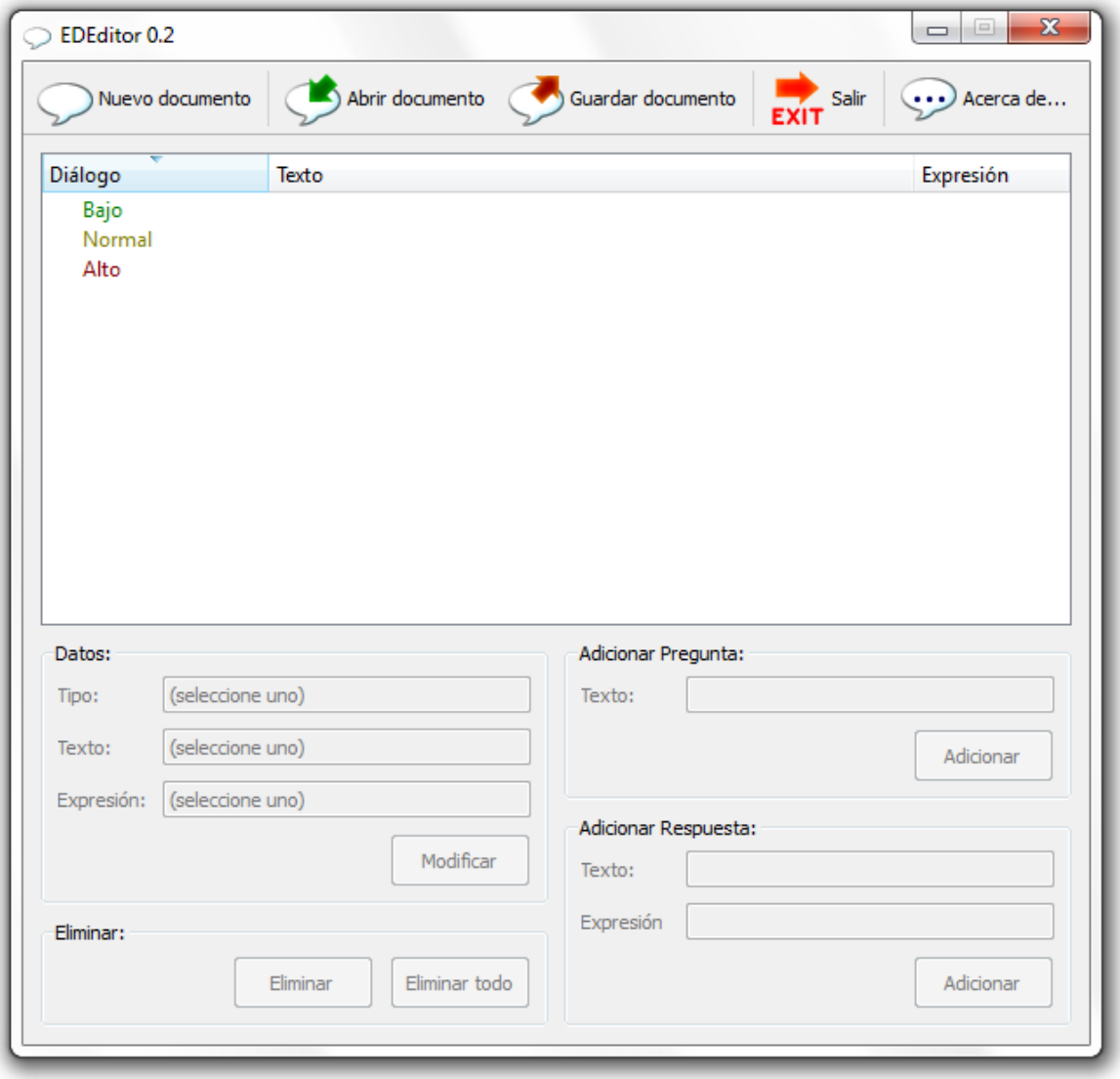

Figura 17: Interface del módulo EDEditor al iniciar.

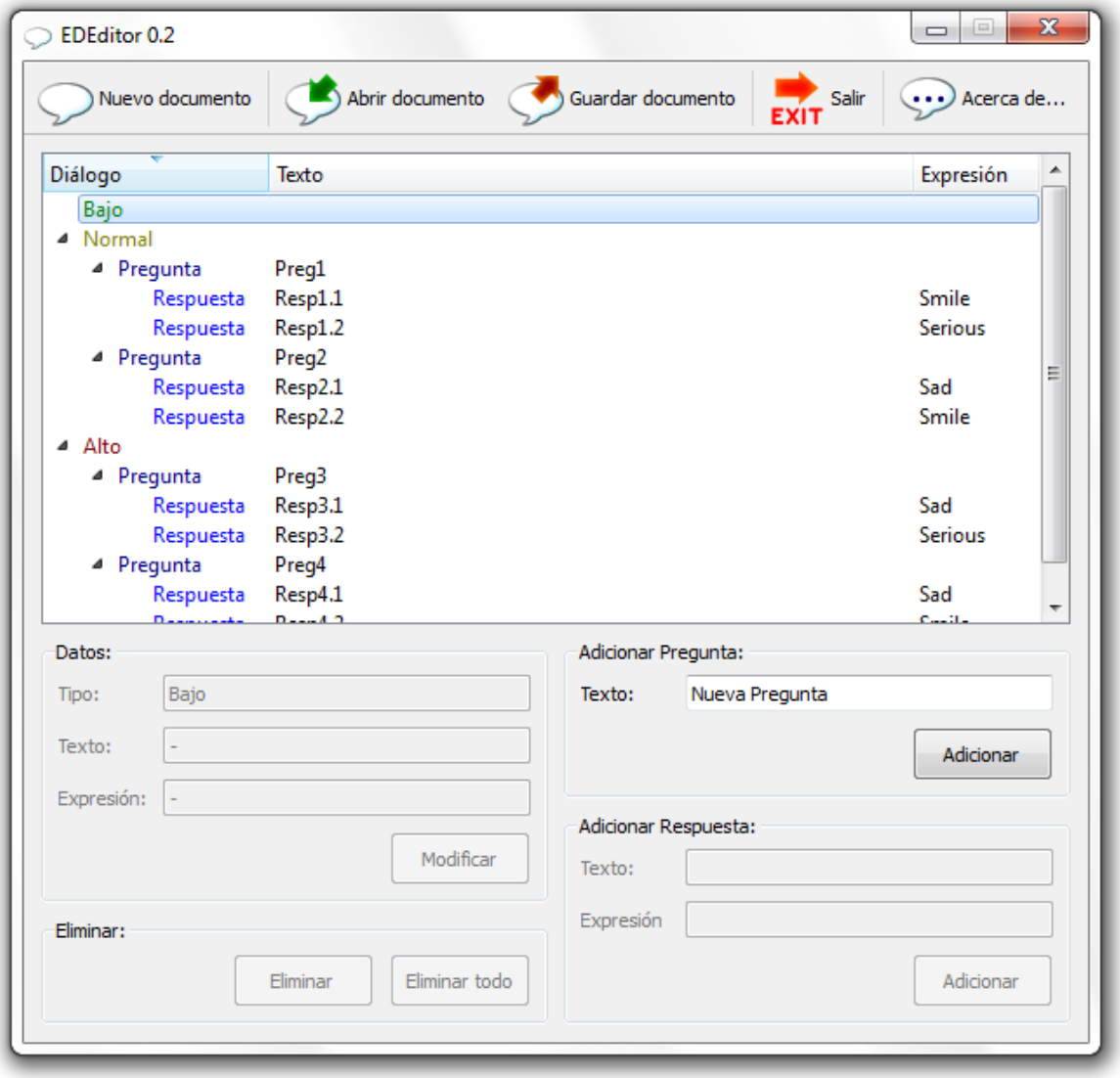

Figura 18: Interface del módulo EDEditor con un archivo de diálogos abierto y listo para agregar una nueva Pregunta.

PROPUESTA DEL MÓDULO EDCORE.

La captura siguiente fue tomada en el **Sistema de Entrenadores Aduaneros** usando el sistema de diálogos desarrollado para esa aplicación.

Figura 19: Uso del módulo EDCore en una entrevista. Interface donde el usuario selecciona que Pregunta realizarle al pasajero.

CONCLUSIONES GENERALES.

Se logró desarrollar los dos módulos que permiten la **gestión de diálogos** requerida por el **Sistema de Entrenadores Aduaneros** siendo necesario destacar que:

- Se propuso una modificación del tipo de diálogo: **basado en estados finitos** para lograr el nivel de simplicidad que necesita el **Sistema de Entrenadores Aduaneros**, utilizando en vez de estados interconectados por nodos, variables en una estructura en forma de árbol.
- El uso de la interface de programación DOM posibilitó no tener que cargar los archivos de diálogo cada vez que se necesiten. Al tenerlos en memoria se puede navegar en cualquier dirección dentro del árbol que contiene los diálogos y así se logra menor tiempo de acceso a la información almacenada en memoria.
- El uso del sistema de diálogos propuesto responde a las necesidades del **Sistema de Entrenadores Aduaneros**, agilizando el trabajo de programadores y editores en la gestión de la información almacenada en los archivos de diálogos.
- Al encontrarse ambos módulos acoplados al **Sistema de Entrenadores Aduaneros** se puede asegurar que se obtuvo un producto totalmente funcional.

RECOMENDACIONES.

Se recomienda lo siguiente:

- Ampliar las funcionalidades del sistema de diálogos propuesto para lograr que no solo sea posible utilizarlo en el **Sistema de Entrenadores Aduaneros**.
- Para futuras actualizaciones desarrollar una variante que utilice las bibliotecas estándar de C++.

BIBLIOGRAFÍA.

Alwood, J. 1992. *On Dialogue Cohesion.* Dept. of Linguistics, University of Guthemburg. 1992.

Bates, J. 1994. *The role of emotion in beliavable agents.* 1994.

Brownell, David. 2008. *SAX2.* s.l. : O'Reilly, 2008. ISBN: 0-596-00237-8.

Brusk, Jenny. 2006. *Game Dialogue Management.* 2006.

Brusk, Jenny y Björk, Staffan. 2009. Gameplay Design Patterns for Game Dialogues. *Breaking New Ground: Innovation in Games, Play, Practice and Theory.* s.l. : Authors & Digital Games Research Association (DiGRA), 2009.

Brusk, Jenny, y otros. 2007. *DEAL - Dialogue Management in SCXML for Believable Game Characters.* 2007.

Bunt, H. 1994. *Context and dialogue control.* s.l. : Think Quarterly, 1994.

Byörk, Staffan y Holopainen, Jussie. 2005. *Patterns in Game Design.* s.l. : Charles River Medi, 2005. ISBN 1-58450-354-8.

Campistruz, Jaime González. 2009. *Sistema de Entrenadores Aduaneros: Indicios Aduaneros.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2009.

Cassell, J., y otros. 2000. *Embodied Conversational Agents.* s.l. : The MIT Press, 2000.

Derrberry, Anne. 2005. Definition of Serious games. *Adobe.* [En línea] 2005. http://www.adobe.com/resources/elearning/pdfs/serious_games_wp.pdf.

Electronic Arts News. 2010. *Electronic Arts.* [En línea] 2010. http://ea.com/news/me2#.html.

Graells, Dr. Pere Marqués. 2001. *Los videojuegos: Las claves del éxito.* s.l. : Pangea, 2001.

Gratch, J., y otros. 2002. *Creating interactive virtual humans: Some assembly required.* Universidad de Pennsylvania. 2002. Reporte técnico, Sistemas Inteligentes IEEE.

Hégaret, Phillippe Le. 2002. The W3C Document Object Model (DOM). *World Wide Web Consortium.* [En línea] 2002. http://www.w3.org/2002/07/26-dom-article.html.

Jufarsky, D. y Martin, J. H. 2009. *Speech and language processing.* s.l. : Pearson International, 2009. ISBN 978-0-13-504196-3.

Krebs, Jochen. 2007. Library: IBM. *IBM.* [En línea] Enero de 2007. http://www-128.ibm.com/developerworks/rational/library/jan07/krebs/index.html.

Mc Tear, M. F. 2002. Spoken Dialog Technologies: Enabling the conversational user interface. *ACM Computing Surveys.* 2002.

Means, W. Scott y Bodie, Michael A. 2008. *The Book of SAX.* s.l. : No Starch Press, 2008. ISBN: 1-886411-77-8.

Ron, Jeffries. 2001. What is Extreme Programming? *Extreme Programming Magazine.* [En línea] 2001. http://www.xprogramming.com.

Simón, Pablo Iglesias. 2005. *El diseñador de sonido: Función y esquema de trabajo.* s.l. : ADE-Teatro, 2005.

Ulrich, Michael. 1985. *Atlas de música.* s.l. : Alianza, 1985. ISBN 84-206-6999-7.

USPTO. 1995. *Television gamming apparatus and method.* EUA, 1995. Software.

W3C. 2011. *World Wide Web Consortium.* [En línea] 2011. http://www.w3.org/TR/REC-xml.

Wells, Don. 2009. Extreme Programming: A gentle introduction. *Extreme Programming.* [En línea] 2009. http://www.extremeprogramming.org.