Universidad Carlos III de Madrid

Escuela Politécnica Superior

Departamento de Informática

Ingeniería en Informática

Proyecto Fin de Carrera

Herramienta para el Análisis y Anotación de Conocimiento Pragmático para Modelos de Diálogo

Autor: Esperanza Albacete García

Tutores: Francisco Javier Calle Gómez, María Dolores Cuadra Fernández

Agradecimientos

En primer lugar, tengo que agradecer a mis padres todo lo que se han esforzado para que desde pequeña pudiera estudiar y ahora mismo me encuentre a punto de terminar la carrera. Sin sus gestos de ánimo y apoyo cuando las cosas me iban mal nunca lo hubiese conseguido. Gracias a mi hermano Tomás y a Julia: me habéis enseñado muchas cosas, una de ellas, a no rendirme.

Muchas gracias a Javi por esos '5 minutillos' que me has dedicado siempre que lo necesitaba para resolverme cualquier duda y a Loli porque además de explicarme ponías la chispa alegre a las reuniones. Habéis sido unos tutores estupendos.

También tengo mucho que agradecerles a mis amigos de clase, los “Cebollitas” porque durante la carrera habéis hecho que todos esos días de prácticas, de exámenes y de agobio hayan sido más llevaderos. Me lo he pasado y sigo pasándomelo genial con vosotros. Gracias a mis compañeros del Labo por haberme aceptado tan bien desde el primer momento y por crear tan buen ambiente de trabajo.

Para terminar, no me gustaría despedirme sin dedicarle unas líneas a la persona que ha hecho de estos años de carrera los mejores de mi vida y que hace que me levante feliz cada día: gracias Kike.

Índice de contenidos

Índice de ilustraciones

1 Introducción

La expansión de las nuevas tecnologías acarrea la necesidad de mejorar las interfaces de comunicación entre humanos y máquinas. Una de las mejoras más perseguidas es conseguir esta interacción sin que las personas tengan que adquirir previamente conocimientos específicos o realizar un aprendizaje técnico antes de manejar cualquier máquina. La interacción natural está enfocada a imitar el comportamiento de la interacción entre humanos.

1.1 Marco de trabajo y motivación

Con el fin de imitar el comportamiento humano es necesario estudiarlo previamente y extraer el conocimiento a partir de la comunicación entre dos personas. Este conocimiento extraído debe formalizarse de algún modo para que el sistema de interacción sea capaz de utilizarlo.

Es necesario distribuir todo el conocimiento que se obtiene de las interacciones en distintos modelos de conocimiento, cada uno de ellos especializado en un aspecto concreto de la interacción. Suelen clasificarse en función de la naturaleza de su conocimiento o el dominio al que hacen referencia y forman parte de la arquitectura cognitiva de un sistema de interacción natural.

En la Ilustración 1.1 se muestra un esquema de la arquitectura cognitiva de un sistema de interacción natural propuesta por (Calle, 2004). En este esquema se pueden apreciar diferentes módulos, cada uno de ellos independiente entre sí pero relacionado con los demás.

El Modelo de Presentación trabaja junto con los componentes de interfaz (reconocedor de voz, interfaz de usuario, avatar, etc.) con el propósito de coordinar todas las entradas de información en la fase de interpretación y todas las salidas en la fase de generación para así poder producir intervenciones completas y coherentes. Además ayuda a gestionar los turnos y manejar el estado de la interacción.

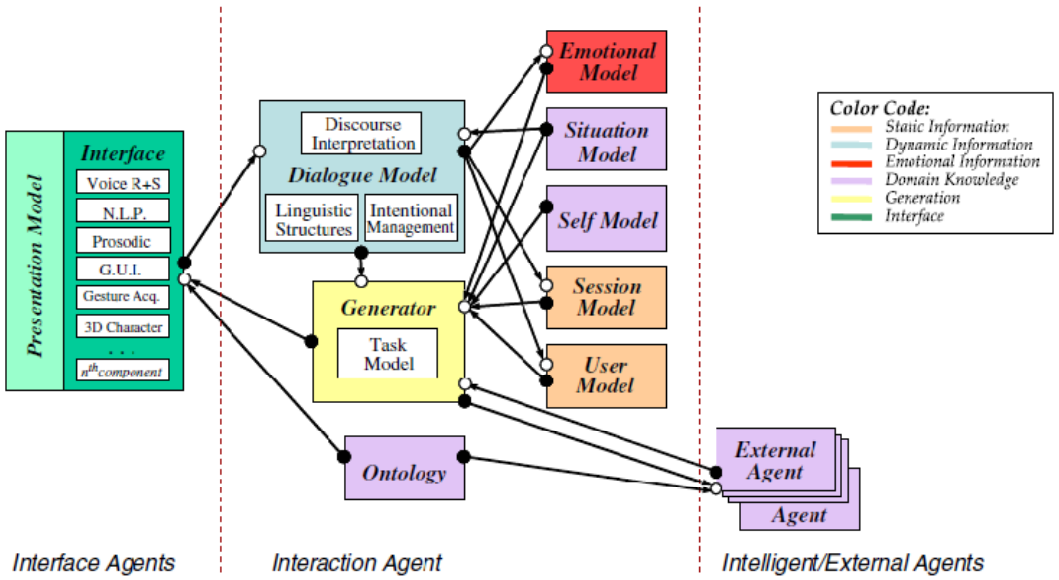

Ilustración 1.1: Arquitectura cognitiva

El componente Interfaz representa la cara visible para el usuario. Su función es recoger las entradas por parte del usuario y devolver las salidas por parte del sistema. Este módulo puede ser implementado mediante diversos tipos de interfaces, como puede ser un reconocedor de gestos, un avatar 3D, un reconocedor de voz, un procesador de lenguaje natural, etc.

El Modelo de Diálogo fija el estado de la evolución de la interacción y define el comportamiento interactivo del sistema. Para ello, procesa las acciones comunicativas del interlocutor y desarrolla las suyas propias.

El Generador de Discurso, que contiene un Modelo de Tareas, es el encargado de gestionar las funciones o tareas que el sistema debe llevar a cabo en cada momento de la interacción (por ejemplo, consultar un reloj interno o conectarse a internet) y producir el contenido de las interacciones del sistema.

La Ontología se encarga de gestionar el conjunto de conceptos y sus relaciones que serán susceptibles a ser referenciados a través de términos dentro de un dominio de interacción.

El resto de modelos se encargan de gestionar otros aspectos de la interacción, estos son: Modelo Emocional (opera sobre información referente a emociones), Modelo de Situación (define las circunstancias en las que se enmarca la interacción), Automodelo (gestiona metas individuales del sistema forjando una personalidad), Modelo de Sesión (almacena el punto de partida y la evolución de la interacción) y Modelo de Usuario (maneja los datos del interlocutor).

Todos estos componentes forman parte de una arquitectura multi-agente, lo que implica que se ejecutan en procesos diferentes, actúan de modo independiente y colaboran entre ellos persiguiendo un mismo objetivo.

Para que esta arquitectura consiga realizar una interacción lo más natural posible se requiere conocimiento específico de un dominio concreto. Este conocimiento proviene de un corpus de un dominio específico que es adquirido a través de la observación de interacciones entre humanos. En esta observación no sólo se tendrán en cuenta los diálogos que suceden, sino que además se recogerán datos gestuales, cambios de tono, expresiones faciales, etc.

Toda la información del corpus adquirido debe ser analizada y anotada por expertos en lingüística, concretamente pertenecientes al área de la pragmática. De esta forma, se hace posible que la información contenida en el corpus pueda ser interpretada por el sistema y además sea utilizada para conseguir imitar el comportamiento humano.

1.2 Objetivos

Este proyecto se engloba dentro del marco del modelo de diálogo y tiene como objetivo desarrollar una herramienta que facilite la anotación del conocimiento pragmático de corpus. La finalidad que se persigue es la implementación de una herramienta fácil de manejar e intuitiva que permita a usuarios expertos en pragmática llevar a cabo todas las fases de las que consta el análisis individual de una muestra de diálogo.

La herramienta estará orientada a la anotación de muestras de diálogo de modo independiente, si bien parte del conocimiento anotado puede ser reutilizado en sucesivas anotaciones. Esto quiere decir que la herramienta Cognos.Dial se encarga de facilitar el análisis individual de cada muestra de diálogo y de almacenar la información recogida para cada uno de los análisis en una base de conocimiento. Sin embargo, este conocimiento no puede ser utilizado directamente por el modelo de diálogo, sino que se requiere realizar una unificación del conocimiento individual.

Para que el conocimiento anotado pueda ser utilizado por el modelo de diálogo, es necesario que sobre las muestras individuales analizadas se apliquen algoritmos de unificación (tanto de tareas como de segmentos) y un algoritmo de aprendizaje para el compromiso. El conocimiento unificado se almacenará en una base de conocimiento global con el propósito de ser utilizada en el modelo de diálogo. Este conjunto de funciones que operan globalmente sobre la totalidad de las muestras de un corpus quedan fuera del ámbito del proyecto pero serán implementadas en un futuro próximo.

2 Estado del arte

En este apartado de la memoria se describirán en primer lugar los conceptos teóricos fundamentales que permitirán al lector comprender los restantes apartados de la memoria. En segundo lugar se dará un repaso al trabajo previo realizado que sirve como base para el desarrollo de la herramienta y, por último, se realizará un estudio de las herramientas relacionadas analizando las diferencias existentes entre ellas y justificando la necesidad de desarrollar Cognos.Dial.

2.1 Fundamentos teóricos

Puesto que la finalidad del proyecto es desarrollar una herramienta capaz de facilitar la anotación pragmática de corpus de diálogo, los términos que aparecerán a lo largo de la memoria son de ámbito lingüístico y concretamente se engloban dentro del área de la pragmática. Debido a que se trata de terminología muy específica es necesario introducirla y describirla previamente para comprender los siguientes apartados de esta memoria. Esta fundamentación se complementa con algunas definiciones que el lector podrá encontrar en el glosario de esta memoria.

2.1.1 Actos de Habla y Actos Comunicativos

La teoría de los actos de habla, tal y como la formuló el filósofo del lenguaje J. L. Austin es una de las consecuencias de la filosofía del lenguaje peculiar. Aparece formulada en su obra póstuma *How to Do Things with Words* (Austin, 1962). Dicha teoría es el arranque de uno de los enfoques más populares en el área de la pragmática.

El efectuar un acto de habla, expresando una oración correcta gramaticalmente y con sentido, implica un compromiso con el entorno. Un acto de habla puede ser solicitar información, ofrecerla, disculparse, expresar indiferencia, expresar agrado o desagrado, amenazar, invitar, rogar, etc.

Un acto de habla o acto ilocutivo es un tipo de acción que involucra el uso del lenguaje natural y está sujeto a un cierto número de reglas convencionales. Austin presenta una propuesta de agrupación de estos actos:

- **Acto locutivo:** Es la acción de realizar una expresión oral, es decir, lo que se dice.
- **Acto ilocutivo:** Es la intención o finalidad concreta del acto de habla.

- **Acto perlocutivo:** es el (o los) efecto(s) que el enunciado produce en el receptor en una determinada circunstancia.

Se podría concluir de esto que en toda acción comunicativa oral estarán presentes estos tres tipos de actos de forma simultánea. También existe comúnmente otra división de actos de habla: actos directos y actos indirectos. Los actos directos son aquellos en los que el aspecto ilocutivo literal de la oración coincide con el aspecto ilocutivo real, es decir, se expresa exactamente la intención del hablante. Los actos indirectos son aquellos en los que esta coincidencia no se produce, es decir, son las frases en las que la finalidad es distinta de lo que se expresa directamente.

En toda interacción humana no solo intervienen acciones de tipo verbal (tanto oral como escrito). Por ello, con el fin de conseguir una mayor generalidad de información y por tanto recoger factores no verbales de la interacción, se introduce el concepto de Actos Comunicativos. Como extensión de los anteriores, permiten unificar cualquier expresión (verbal y no verbal) en la misma representación y con el mismo formato, por tanto permitirán representar gestos, pausas, solapamientos y otro conocimiento prosódico que todo humano identifica y representa en sus interacciones.

2.1.2 Dominio de interacción

Dominio de interacción es un concepto fundamental en cualquier sistema de interacción natural. Se define dominio como un área de experiencia y conocimiento en alguna actividad del mundo real. Interacción es el conjunto de acciones e intercambios de información que se suceden entre varios sujetos, objetos funciones, etc. En el caso particular de este proyecto la interacción a la que se refiere este documento constantemente es la comunicativa. Un dominio de interacción determina por tanto el conocimiento necesario para que se produzcan los intercambios entre ambos interlocutores y que éstos sean satisfactorios para un conjunto de fines.

Es necesario discernir el dominio de cada interacción ya que no todas las personas poseen experiencia y conocimiento para interactuar en determinadas áreas. Todo individuo está capacitado para conversar en multitud de dominios pero nunca en todos, pues siempre llega un punto en el cual se requiere más conocimiento para ampliar el conjunto de dominios y sin éste conocimiento un individuo en un dominio que le es ajeno será incapaz de transmitir opiniones, actuar o interpretar.

Análogamente a las personas, sucede lo mismo a los sistemas de interacción natural. Es necesario limitar el dominio para que el sistema se desenvuelva satisfactoriamente (y aún así no siempre se consigue). Por ello, para alimentar a estos sistemas, este proyecto se orientará a suministrar conocimiento a cada dominio de interacción concreto mediante una interfaz general para ello.

Un dominio de interacción contendrá un conjunto de escenarios que representan distintos tipos de conversaciones que se puede dar entre el usuario y el sistema de

interacción con un determinado objetivo o una tarea específica. Cada uno de estos escenarios puede dar lugar en un corpus a varios diálogos que demuestran fielmente cómo un humano realizará dichas tareas con otro humano experto del dominio.

Además, un diálogo proveniente de cualquier interacción está compuesto por intervenciones, que identifican los distintos turnos por parte de los participantes sin que presenten interrupciones por parte del segundo hablante. Cada una de estas intervenciones puede constar de una o varias frases así como de distintos símbolos no verbales.

2.1.3 Modelos de diálogo

En las últimas décadas se han llevado a cabo numerosas investigaciones en el ámbito de la interacción persona-ordenador y de ellas han surgido los sistemas de diálogo.

Un sistema de diálogo debe ser capaz de adquirir, analizar y procesar las expresiones que realiza el usuario de un modo natural, como si las dijera otra persona (Calle, 2004). Dentro del estudio de los sistemas de diálogo, se han desarrollado varias propuestas y clasificaciones. Según (D. Scott, 1997), se pueden clasificar de la siguiente forma: coherencia discursiva (Hobbs, 1985), esquemas (McKeown, 1985), teoría de la estructura retórica (W. C. Mann, 1987) y atención, intención y frases (B. Grosz, 1986). Otra posible clasificación de (McTear, Jokinen, & Larson, 2008), que extiende la propuesta por (P. R. Cohen, 1997) distingue cinco categorías:

- **Gramáticas de diálogo** (J. A. Levin, 1977)**:** Se basan en los juegos de diálogo, cuyo fundamento teórico se encuentra en los juegos de lenguaje de Wittgenstein formulados en (Wittgestein, 1988). Juego de Lenguaje es el término que hace referencia a las secuencias de actividades lingüísticas que los humanos pueden desarrollar al interactuar. La implementación de los juegos de lenguaje como un conjunto de pares de adyacencia se denomina Juegos de Diálogo. El conjunto de diálogos válidos se restringe al determinar los posibles pares de adyacencia (o parejas de actos comunicativos) que pueden sucederse unos a otros. Este tipo de gramáticas ofrecen muy buen rendimiento en dominios muy rígidos y reducidos, no obstante necesita de una gran cantidad de corpus para alcanzar resultados satisfactorios.
- **Modelos basados en planes:** Se basan en la idea de que los humanos elaboran planes con el fin de alcanzar sus objetivos. Si se parte del hecho de que un diálogo se establece con el fin de alcanzar una meta, cualquier interacción puede ser vista como el desarrollo de una serie de acciones planificadas previamente que tienen como meta alcanzar un objetivo.

- **Marcos y Estados de Información** (R. Cooper, 1998)**:** Este modelo se encuentra orientado a la practicidad del diálogo. Un marco o estado de información tiene definidos un conjunto de requisitos que es necesario satisfacer para realizar la transición al siguiente marco.
- **Procesamiento Intencional**(B. Grosz, 1986)**:** Este modelo de diálogo organiza la estructura del discurso según el foco de atención, las metas o la intención del emisor y la estructura de las secuencias lingüísticas.
- **Modelos de acción combinada** (Clark, 1996)**:** Una acción combinada presenta una serie de características: la necesidad de participantes que desempeñan determinados roles, la existencia de unas metas individuales para cada uno de los participantes y la existencia de un conjunto de metas compartidas para todos los participantes (es decir, aquella que todos aspiran a alcanzar). El conjunto de metas compartidas se denomina zona común. El conocimiento mutuo de un hecho (X) se define, para dos participantes A y B por las siguientes reglas: a) A conoce X**,** b) B conoce X, c) Ay B conocen a), b) y c) y así sucesivamente. De este modo, un participante aceptará el compromiso para desarrollar las metas compartidas siempre que crea que el resto de participantes lo ha asumido también

2.2 Trabajo previo

Durante los últimos diez años, el grupo (LaBDA) de la Universidad Carlos III de Madrid ha estado trabajando en el área de la Interacción Natural y entre sus tareas de investigación ha desarrollado un modelo de diálogo de acción combinada específico denominado Modelo de Hilos. El resultado generado para este modelo es extensible a los demás modelos de diálogo puesto que el conocimiento que utilizan los modelos de acción combinada es un superconjunto del necesario para los demás tipos de modelado de diálogo (definidos en el apartado 2.1.3).

El Modelo de Hilos pertenece a la categoría de modelos de diálogo basados en procesamiento intencional y acción combinada. Este modelo aporta los mecanismos necesarios para organizar, interpretar, y planificar diálogos según las intenciones que manifieste el usuario en sus intervenciones y a las propias del sistema. Las intenciones surgirán motivadas por el resultado de la ejecución de una tarea (externa o interna) requerida para satisfacer un hilo de usuario o por una programación de hilos (hilos planificados, o resultado de tareas internas a la interacción). Por tanto, este modelo se engloba en la categoría de diálogo de 'iniciativa mixta'.

Los beneficios del modelo de hilos son, básicamente, dotar al sistema de interacciones más flexibles, de mayor coherencia y que el usuario desarrolla con mayor

comodidad, aumentando la probabilidad de terminar exitosamente las metas. Algunas de sus características más notables de este modelo son:

- **Gestión coherente y flexible del diálogo:** gracias al procesamiento intencional y a la gestión de la sintonía.
- **Disminución de la necesidad de corpus:** por la gestión de unidades funcionales pequeñas.
- **Implicación del sistema en el éxito del diálogo** por la gestión del compromiso y la sintonía, produciendo una interacción más cómoda para el usuario.
- **Progreso ágil de la interacción**: manifiesta por incrementos en el número de pasos realizados por turno, y en el número de metas satisfechas por turno.
- **Gestión de los espacios contextuales:** la organización intencional determina los ámbitos contextuales, y los vínculos entre ellos.
- **Separa las aportaciones** de usuario y sistema al diálogo, posibilitando futuras revisiones.
- **Establecimiento de distintas estrategias** para desarrollar un hilo.
- **Depuración de partes del diálogo:** al posibilitar retomar en cualquier momento un hilo anterior, ya estuviera pendiente de resolución o resuelto (para su refinamiento).
- **Introducción de iniciativas** originadas por distinto tipo de conocimiento, y que pueden incluso iniciar el diálogo completo.
- **Control de iniciativas** para decidir qué iniciativas del sistema son convenientes, y cuáles deben ser desestimadas, dependiendo de la circunstancia y de la *criticidad* de la iniciativa.
- **Juegos de pausas** que manejan la ausencia de intervenciones del usuario como caso particular de expresión, logrando un comportamiento más proactivo.

Este Modelo ha sido llevado a la práctica con anterioridad en numerosos proyectos de adjudicación competitiva (europeos y nacionales): Advice, VIP_Advisor, IntegraTV4ALL, Sopat y Thuban.

2.3 Herramientas relacionadas

Cognos.Dial surge de la necesidad de agrupar en una única herramienta todas las fases que componen la anotación pragmática de corpus lingüísticos. Hoy en día existen aplicaciones que permiten realizar determinadas fases de la anotación, pero hasta el momento ninguna herramienta cubre la totalidad tareas necesarias para anotar todo el conocimiento pragmático de un corpus de diálogo.

Debido a que no existen herramientas similares a Cognos.Dial, no es posible realizar un estudio comparativo de toda su funcionalidad. No obstante, existe una herramienta (Transcriber) que tiene relación con la primera etapa de anotación pragmática (pre-segmentación, es decir división y anotación de los roles de un diálogo), por consiguiente en el apartado 2.3.1 se realizará un análisis de esta aplicación y se compararán algunos detalles de funcionalidad.

Aunque queda fuera del ámbito de este proyecto, hay que señalar que al finalizar todas las fases de anotación en la herramienta, será posible exportar el análisis a un fichero XML que seguirá el formato definido en (D. Cuadra, 2009). Para fundamentar esta extensión, en este apartado también se incluirá un estudio sobre esquemas de anotación pragmática similares, como son DAMSL y sus extensiones.

2.3.1 Transcriber

La herramienta Transcriber tiene como objetivo facilitar la anotación manual de señales de voz. Para ello, proporciona una interfaz gráfica que permite realizar la segmentación de grabaciones de voz de larga duración así como su transcripción. Además permite realizar el etiquetado de los turnos de cada participante, así como los cambios de tema y de condiciones acústicas. Por tanto, la funcionalidad que ofrece esta herramienta se corresponde en gran parte con la requerida en la fase de pre-segmentación del proyecto Cognos.Dial a excepción del proceso de transcripción automática del diálogo desde un fichero de audio.

Transcriber está específicamente diseñado para la anotación de las grabaciones de la emisión de noticias y para la creación de corpus utilizados en el desarrollo de sistemas automáticos de transcripción de noticias, pero sus características pueden ser útiles en otras áreas de investigación del habla.

La aplicación ha sido desarrollada en el lenguaje de programación Tcl/TK y extensiones del lenguaje C. Se basa en la extensión de sonido Snack (que permite el soporte para los formatos de audio más comunes) y en el generador de analizador léxico tclex. La herramienta se distribuye como software libre bajo licencia GNU, y tiene soporte para las plataformas Windows XP/2k, Mac OS X y varias Linux.

Transcriber permite la anotación de varios niveles de segmentación: una segmentación de base para la transcripción ortográfica (por ejemplo, en cada frase, o en cada respiración), la segmentación de los turnos de habla (cuando interviene un nuevo portavoz) y la segmentación de la sección (cuando tiene lugar un cambio de tema). Además mediante esta herramienta es posible realizar la segmentación cuando hay sonido de fondo en el diálogo. En el caso de Cognos.Dial sólo se realizará una segmentación en los turnos de los participantes, anotando para cada intervención el rol del hablante que la lleva a cabo.

Como se muestra en la Ilustración 2.1, Transcriber consta de una pantalla de segmentos bajo la señal y en el editor de texto. Mediante un menú contextual se puede activar la visualización de cada uno de los segmentos en la señal.

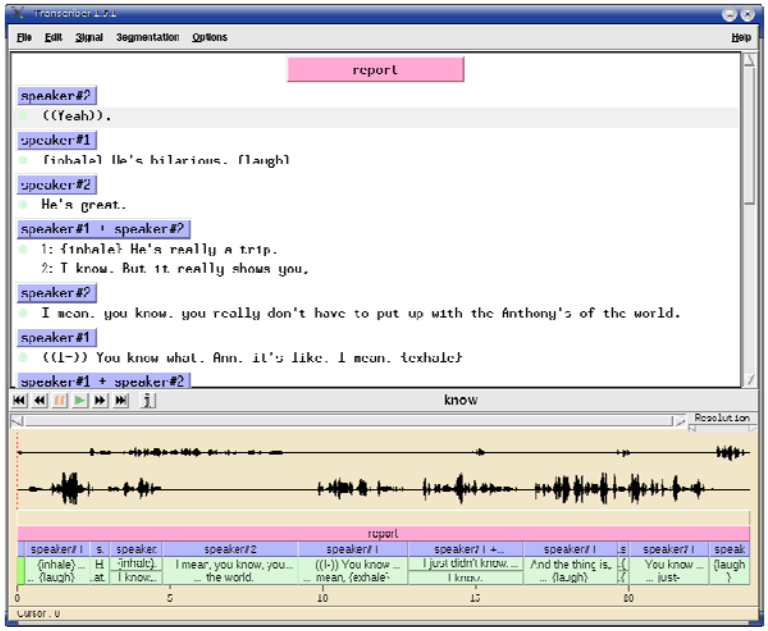

Ilustración 2.1: Interfaz gráfica de Transcriber

En resumen, Transcriber se presenta como una herramienta potente para la realización de la transcripción y de la pre-segmentación, al permitir la traducción automática del texto desde un fichero de audio e incluir más elementos de anotación aparte de la división en roles del diálogo. Sin embargo, estas características no son esenciales para realizar el proceso de anotación pragmática de un corpus y el hecho de incluirlas supondría una gran carga extra de trabajo en el proyecto, que impediría alcanzar otros objetivos más necesarios. No obstante, se tomarán en cuenta para posibles ampliaciones futuras de la herramienta Cognos.Dial.

2.3.2 DAMSL (Dialog Act Markup in Several Layers)

DAMSL consiste en un esquema de anotación pragmática desarrollado por el Multiparty Discourse Group en la Iniciativa de Investigación del Discurso (DRI) en el año 1996. Este esquema fue desarrollado en un principio para diálogos de dos participantes que colaboran para resolver un determinado problema.

Un diálogo se divide en unidades denominadas turnos, en los cuales un hablante tiene temporalmente el control del diálogo e interviene durante un período de tiempo. Dentro de un turno, la intervención del participante puede dividirse en varias unidades de expresión. En el esquema (DAMSL) el término expresión está basado en un análisis de las intenciones del hablante y cada una de ellas se podrá clasificar (según unas etiquetas de expresión definidas) para registrar la unidad de propósito y la función de la misma en el diálogo.

Las etiquetas de expresión, indican algún aspecto particular de la unidad de expresión en sí misma y resumen las intenciones del hablante y el contenido de la expresión. Estas etiquetas se clasifican en cuatro categorías principales:

- **Communicative Status:** indica si la expresión es comprensible y si se completó con éxito.
- **Information Level:** caracterización del contenido semántico de la expresión.
- **Forward Looking Function:** indica cómo la expresión actual limita las acciones futuras de los participantes y afecta al discurso.
- **Backward Looking Function:** indica cómo la expresión actual se relaciona con el discurso anterior.

Como se puede apreciar, el esquema DAMSL es bastante extenso y recoge gran cantidad de información pragmática. Sin embargo, esta información presenta un análisis superficial mientras que los modelos de acción combinada precisan de un análisis profundo. Por ello, se ha decidido seguir un esquema más específico y adaptado a las necesidades del modelo propio. Dicho esquema se encuentra definido en el artículo (D. Cuadra, 2009). De todos modos, la exportación del análisis a un fichero con formato DAMSL no se descarta y se presenta como una línea futura del proyecto.

2.3.3 Extensiones de DAMSL

Del esquema de anotación DAMSL han surgido numerosas extensiones y especializaciones. Una de ellas es Switchboard-DAMSL (D. Jurafsky, 1997) desarrollada en la Universidad de Colorado, EEUU. El dominio que cubre el esquema es de ámbito general basado en conversaciones telefónicas entre extraños y ha sido desarrollado para la investigación del habla coloquial y del lenguaje espontáneo. En esta extensión del esquema de anotación DAMSL se han introducido nuevas etiquetas con el fin de recoger una mayor cantidad de información pragmática. Concretamente, las extensiones se han realizado dentro de las categorías Communicative Status, Forward Looking Function y Backward Looking Function.

ADAM (R. Cattoni, 2001) es otra versión modificada del esquema de etiquetado DAMSL y comparte con éste las mismas características utilizadas para capturar la dimensión comunicativa de un turno de diálogo. La diferencia principal reside en que el esquema ADAM permite la práctica de la anotación pragmática de tres capas: en la primera capa cada turno de diálogo es caracterizado con respecto a su nivel comunicativo; en la segunda capa la anotación capta la dimensión ilocutiva de la expresión o expresiones incluidas en el turno; en la tercera capa se caracterizan las relaciones en el discurso entre diferentes expresiones.

3 Especificación de requisitos

En este apartado se describirán los requisitos del software a un nivel de detalle suficiente para diseñar un sistema que los satisfaga por completo. Tal y como detalla el estándar IEEE 830-1998, (The Institute of Electrical and Electronics Engineers, 1998), una buena especificación de requisitos ha de ser:

- Correcta. Cada requisito declarado debe encontrarse en el software reflejando las necesidades reales del usuario.
- Inequívoca. Cada requisito declarado tiene solo una interpretación, para ello se evitará usar palabras o descripciones ambiguas que puedan dar lugar a confusión.
- Completa. En esta especificación estarán presentes todos los requisitos relacionados con la funcionalidad que el sistema va a proporcionar.
- Consistente. Ninguno de los requisitos o conjunto de ellos presentes en el documento ha de generar conflicto con otros.
- Comprobable. Cada requisito ha de ser fácilmente verificable, esto significa que para cada requisito será posible realizar una prueba que lo valide.
- Modificable. La estructura y estilo de la especificación de requisitos facilitarán que se pueda realizar cualquier cambio en los requisitos de forma consistente.

En este apartado se especificarán los requisitos software definidos para el proyecto Cognos.Dial y a continuación se estudiará la viabilidad del proyecto, en el cual se realizará un análisis de las características del sistema con vistas a verificar si éstas se pueden cubrir. En el último punto se especificarán pruebas de verificación sobre los requisitos que permitirán comprobar si el producto final satisface cada uno de ellos.

3.1 Especificación de requisitos software

En este apartado se enumeran y se explican brevemente los requisitos extraídos durante las diferentes reuniones con el usuario. Estos se dividen en requisitos generales y en específicos. Los primeros son los que detallan el propósito de la herramienta así como los requisitos de interfaces ya sea con el usuario o con otros sistemas. Los requisitos específicos describen más a fondo las capacidades que debe de tener la herramienta.

3.1.1 Requisitos generales

A continuación se realizará una descripción de los requisitos generales que presentan a alto nivel el objetivo principal de la herramienta y la funcionalidad que debe cubrir.

Requisito 1: Se desarrollará una herramienta capaz de auxiliar en el análisis y anotación pragmática de corpus de diálogo.

Requisito 2: La herramienta automatizará algunos procesos mediante un análisis automático y/o supervisado.

Requisito 3: Tras el análisis, la herramienta trasladará el conocimiento obtenido directamente a su base de conocimiento correspondiente (al menos para un modelo de diálogo).

3.1.1.1 Interfaces hardware

Requisito 4: Para ejecutar la aplicación se debe de tener una máquina que disponga como mínimo del siguiente hardware: procesador Intel™ Pentium ® 4 con 1 Gb de memoria RAM y la tarjeta gráfica que requiera el sistema operativo.

Requisito 5: Para almacenar la base de conocimiento se requiere una maquina servidora con las siguientes características BATMAN

Requisito 6: El sistema recibirá la información por parte del usuario mediante un teclado y un ratón. La información de salida se mostrará al usuario a través de una pantalla.

3.1.1.2 Interfaces software

Requisito 7: La aplicación funcionará sobre los sistemas operativos Windows 2000, Windows XP y Windows Vista.

Requisito 8: Es necesario que el equipo tenga instalada la máquina virtual de Java versión 1.6 y posteriores.

Requisito 9: El Sistema Gestor de Base de Datos que almacenará la base de conocimiento será Oracle® 10g y deberá de estar instalado en la maquina servidora.

Requisito 10: Para permitir la ejecución de operaciones sobre la base de datos desde el lenguaje de programación Java es necesario que el equipo soporte el driver de conexión JDBC

3.1.1.3 Interfaces de comunicaciones

Requisito 11: Se requiere que el equipo en el que se ejecuta la aplicación se conecte al servidor de bases de datos mediante una red de área local utilizando para ello el protocolo TCP/IP.

3.1.1.4 Requisitos de rendimiento

Requisito 12: La herramienta debe permitir al usuario realizar el proceso de análisis individual de muestras de forma ágil, de modo que ninguna operación dure más de cinco segundos y en caso de que las haya, vendrán monitorizadas por una barra de progreso.

3.1.1.5 Requisitos de seguridad

Requisito 13: La validación de seguridad se hará contra la base de datos mediante un mecanismo de autenticación. Cada usuario deberá tener un nombre y una contraseña asociada para poder entrar en el sistema.

3.1.2 Requisitos específicos

Las fases de análisis y formalización de corpus lingüístico que se llevarán a cabo en el sistema serán las siguientes:

- **Pre-segmentación:** Permite la división de la muestra en piezas de discurso. Se realiza asignando a cada intervención un rol determinado y fragmentándola después en las piezas correspondientes.
- **Realización temporal:** Permite la inclusión de cierta información prosódica (silencios y solapamientos) entre dos intervenciones.
- **Microanálisis:** Facilita la asignación de uno o más actos comunicativos a cada discurso.
- **Segmentación:** Posibilita la división del diálogo en unidades funcionales abstractas denominadas segmentos.

La interfaz de usuario mostrará de manera independiente cada una de las cuatro fases, guiando al analista a través del análisis y permitiendo regresar a fases anteriores para hacer modificaciones.

En este apartado se describen los requisitos específicos de cada una de las cuatro fases del análisis. Se comenzará explicando aquellos que son comunes a todas ellas y, posteriormente, se describirán los propios de cada fase por separado.

3.1.2.1 Requisitos específicos comunes a las cuatro fases

Requisito 14: La aplicación permitirá crear, describir y registrar nuevos corpus.

Requisito 15: La aplicación permitirá crear un nuevo escenario dentro de un corpus. El nuevo escenario estará definido por una descripción.

Requisito 16: La aplicación permitirá crear una nueva muestra dentro de un escenario perteneciente a un corpus determinado. La nueva muestra contendrá el texto de un diálogo el cual podrá ser vacío o extraído de un fichero externo. En cualquiera de los dos casos, el usuario podrá modificar el contenido del diálogo mostrado.

Requisito 17: La aplicación permitirá abrir una muestra de diálogo almacenada en la base de datos para que el usuario visualice o continúe su análisis.

Requisito 18: El sistema permitirá mostrar y modificar la descripción de un corpus o de un escenario.

Requisito 19: La aplicación permitirá guardar el análisis que ha desarrollado el usuario sobre una muestra. Si el análisis no está completo, se guardará el trabajo parcial del usuario, permitiéndole retomarlo cuando éste desee.

Requisito 20: El sistema tendrá una pequeña franja informativa en la cual se mostrará el último cambio, transacción u operación que se haya llevado a cabo en el sistema.

Requisito 21: El sistema informará al usuario de cualquier advertencia o error mediante una ventana de tipo "pop-up".

3.1.2.2 Requisitos específicos de la fase pre-segmentación.

Requisito 22: La aplicación permitirá obtener el texto de una muestra de diálogo a través de la carga de un archivo de texto externo.

Requisito 23: El sistema permitirá al usuario asignar uno de los dos roles que maneja el sistema a una frase o conjunto de frases de la muestra. Al asignarle un rol a un fragmento de texto, se cambiará el color de la fuente de dicho fragmento.

Requisito 24: El sistema permitirá al usuario chequear la fase de pre-segmentación para comprobar que no quedan fragmentos de texto sin asignar a un rol y que la muestra no se encuentra vacía.

3.1.2.3 Requisitos específicos de la fase microanálisis.

Requisito 25: El usuario debe de poder asignar manualmente o automáticamente actos comunicativos a las intervenciones obtenidas en la fase de pre-segmentación.

Requisito 26: Se debe de disponer sendos enlaces a las aplicaciones *Cognos.CA* y *Cognos.NL*, respectivamente.

Requisito 27: Las intervenciones pueden contener variables de contexto, por lo cual la herramienta debe de permitir añadirlas manualmente si no se ha realizado un análisis automático.

Requisito 28: Para que esta fase sea correcta, todas las intervenciones han de tener al menos un acto comunicativo asociado y este puede contener o no una o más variables de contexto.

3.1.2.4 Requisitos específicos de la fase realización temporal.

Requisito 29: Se visualizará el diálogo de la fase de pre-segmentación en forma de intervenciones. Lo que separa una intervención de otra será un cambio de rol o un punto y aparte. En la interfaz de usuario se mostrará un gráfico con cada

intervención en una línea diferente. En dicho gráfico se podrá visualizar a su vez el instante temporal en el que tuvo lugar cada intervención.

Requisito 30: Cada intervención se visualizará con un color de fondo correspondiente al rol que le ha sido asignado en la fase de pre-segmentación

Requisito 31: La herramienta permitirá añadir silencios antes y/o después de una intervención concreta. Será posible determinar una duración en segundos para cada silencio. La unidad mínima para un silencio que manejará la herramienta será de medio segundo. El usuario será capaz de modificar gráficamente la duración de dicho silencio.

Requisito 32: El usuario puede definir el tipo de un silencio especificando si se trata de un intervalo (o apelativo), un lapso (o anuncio), una pausa o una pausa oralizada (o pausa locutiva). No será obligatorio determinar el tipo de un silencio, en cuyo caso tomará un valor por defecto.

Requisito 33: La aplicación permitirá añadir o eliminar solapamientos entre dos intervenciones contiguas siempre y cuando no pertenezcan al mismo rol. El solapamiento podrá ser físico o no físico. En el solapamiento físico existirá al menos una palabra solapada entre dos intervenciones. El usuario podrá determinar gráficamente en la herramienta las palabras que se solapan. En el solapamiento no físico no habrá ninguna palabra solapada entre dos intervenciones contiguas.

Requisito 34: Dentro de un solapamiento, el usuario podrá definir el tipo del mismo especificando los cambios en el tono que realizan los interlocutores de la primera y segunda intervención respectivamente. El interlocutor de la intervención solapada podrá bajar el tono, mantenerlo, elevarlo o abortar la intervención. Asimismo, el interlocutor de la intervención que solapa podrá bajar el tono, mantenerlo, elevarlo o robar el turno de la intervención solapada.

Requisito 35: En caso de que exista un silencio entre dos intervenciones, al añadir un solapamiento entre las mismas, la herramienta eliminará automáticamente dicho silencio. El tipo de solapamiento que se creará en este caso será de tipo no físico por defecto.

Requisito 36: En caso de que exista un solapamiento entre dos intervenciones, si el usuario añade un silencio entre ambas, la herramienta eliminará automáticamente el solapamiento existente y creará un silencio por defecto de medio segundo de duración.

Requisito 37: La aplicación permitirá al usuario reproducir un archivo de audio, lo cual facilitará el análisis de la realización temporal de las muestras para las cuales existan grabaciones de voz.

3.1.2.5 Requisitos específicos de la fase segmentación.

Requisito 38: Esta fase permitirá dividir el diálogo en un conjunto de unidades funcionales independientes denominadas segmentos. Para ello, la aplicación deberá permitir la creación de un segmento a partir de un conjunto de intervenciones (o de correspondientes actos comunicativos).

Requisito 39: A cada intervención (o conjunto de actos comunicativos correspondientes a la misma) se le podrá asociar una secuencia. Los tipos de secuencia que podrán serle asignados son: presecuencia, apertura, desarrollo, oscurecimiento o abandono, reapertura, cancelación y cierre.

Requisito 40: La interfaz de usuario permitirá visualizar globalmente la estructura de segmentos junto con las intervenciones (o actos comunicativos asociados) y el tipo de secuencia de cada intervención.

Requisito 41: La edición de segmentos incluirá las siguientes operaciones: crear uno nuevo a partir de un conjunto de intervenciones (o actos comunicativos) seleccionado, eliminar un segmento existente y reasignar a un segmento una nueva disposición de intervenciones (o actos comunicativos). Además se incluirá la opción de aumentar o disminuir el nivel un determinado segmento moviéndolo a la izquierda o a la derecha dentro del conjunto de segmentos.

Requisito 42: El segmento de primer nivel será el segmento base, el cual abarcará todo el diálogo y sobre él no se podrá utilizar ninguna de las operaciones de edición descritas en el Requisito 41: .

Requisito 43: Cuando un segmento se descompone en sub-segmentos, se crea un vínculo decomposicional que podrá ser modificado permitiendo a un determinado segmento subir o bajar en la jerarquía de segmentos. El padre de un segmento está definido por el menor segmento que lo contiene completamente. Cuando un segmento sube en la jerarquía, su nuevo padre será el padre de su actual padre. Cuando un segmento baja en la jerarquía, pasa a ser hijo de uno de sus hermanos actuales.

Requisito 44: Cada segmento creado a partir de un conjunto de intervenciones desarrolla una intención que el usuario podrá editar. Cada intención posee un identificador (breve cadena de caracteres) y un conjunto de características. Además debe ser posible la reutilización de una intención previamente editada.

Requisito 45: La herramienta permitirá definir la precedencia (orden de ejecución) de un determinado segmento con respecto a otro segmento identificado previamente. Los tipos de precedencia son: inicio-inicio, inicio-fin y fin-fin. En toda relación de precedencia habrá un segmento antecedente y otro consecuente, de tal forma que no se permitirá la aparición de precedencias recíprocas (si la secuencia “A” es antecedente de “B”, “B” no podrá ser antecedente de “A”).

Requisito 46: A cada segmento se le podrá asociar un conjunto de presuposiciones. Cada presuposición está caracterizada por un nombre y un valor.

Requisito 47: Esta fase incluirá la gestión atencional de los miembros del diálogo. Se permitirá definir los cambios de foco en el diálogo. Para ello, al seleccionar una secuencia de tipo presecuencia apertura o reapertura se podrá definir el tipo de salto: regular o excepcional.

3.2 Estudio de viabilidad

En este apartado se realizará el análisis de las necesidades propuestas en el apartado anterior y se plantearán soluciones que las satisfagan, teniendo en cuenta restricciones económicas técnicas y operativas. Una vez descritas cada una de las alternativas se valoraran en cada caso las fortalezas y debilidades que ofrecen con el fin de seleccionar la más adecuada

3.2.1 Viabilidad del proyecto

Los usuarios a los que está enfocado el proyecto son en su mayor parte lingüistas que analizarán las expresiones para así alimentar el apartado de lenguaje natural de la base de conocimiento Cognos. Estos usuarios posiblemente no estén familiarizados con aplicaciones y entornos de edición complejos y con muchas opciones, por lo cual es necesario especificar un sistema fácil de usar y de aprendizaje ágil. Para ello la interfaz de usuario (UI) deberá ser todo lo intuitiva y amigable posible.

Para la creación de una interfaz como la comentada en el párrafo anterior es necesario un lenguaje con librerías y API's que dispongan de buenas posibilidades gráficas. Un lenguaje orientado a objetos facilitará la encapsulación, modularidad y en definitiva un buen diseño de la aplicación. Por tanto se utilizará el lenguaje de programación Java, que satisface estas necesidades, además es un lenguaje pre-compilado por lo que el tiempo de respuesta será menor, favoreciendo de esta forma la experiencia de usuario.

El sistema estará basado en una arquitectura cliente-servidor por lo cual las especificaciones tecnológicas se dispondrán siguiendo dos enfoques: el enfoque del cliente y el del servidor. Es necesaria esta división ya que los requisitos tanto de hardware como de software para cada uno de ellos divergen bastante.

3.2.1.1 Cliente

Por un lado será necesaria la máquina que actúa como cliente la cual ejecutará la aplicación y permitirá realizar el análisis de corpus. Esta máquina debe de ser capaz de ejecutar aplicaciones Java, para lo cual como requisito software es necesario que esté instalada la máquina virtual Java (JVM).

Como sistema operativo para el cliente será necesario un sistema operativo que ofrezca soporte actual y que disponga de la mayor difusión posible. Las diferentes opciones actuales son utilizar Windows o utilizar Linux. A favor del primero, está la amplia difusión con la que cuenta y que prácticamente la totalidad de usuarios de ordenadores personales lo utiliza. En su contra corre que es un sistema de pago. El sistema operativo podrá ser tanto de 32 como de 64 bits.

Debido al tipo de operaciones que efectúa el motor de inferencia, la carga de datos a memoria principal es más viable; por un lado ya que la velocidad de la memoria principal es inmensamente superior y no depende en absoluto de otros factores como la latencia de red, y por otro, las operaciones comentadas son en base recursiva, por tanto, es común tener que consultar varias veces un valor. Si este no se almacena en memoria principal, el usuario no apreciará un mecanismo ágil de sugerencia y por otro lado, se sobrecargará la red.

Por lo dicho anteriormente, en cuanto a especificaciones hardware para la parte del cliente deberá de ser de una máquina con memoria RAM suficiente como para permitir la carga del conocimiento almacenado para así permitir al motor de inferencia realizar la sugerencia en base a este.

En cuanto a capacidad de procesamiento, la maquina cliente no ha de ser más potente que cualquier ordenador personal común hoy en día. Ya que se trata en su mayor parte de una herramienta de edición, las operaciones de cálculo se reducen a la inferencia en la cual la velocidad de cálculo no es tan determinante como es la de acceso a los datos.

3.2.1.2 Servidor

Esta herramienta, necesita almacenar los resultados de la edición en una base de conocimiento implementada en un sistema gestor de base de datos (S.G.B.D.). Este ha de ser concurrente ya que múltiples clientes realizarán anotaciones al mismo tiempo. También se busca facilidad en la implementación de la comunicación entre la base de datos con la herramienta. Existe la posibilidad de prescindir de un sistema gestor de base de datos e implementar el almacenamiento en ficheros XML, pero esto complicaría de sobremanera la implementación de la aplicación. Se tomará por tanto Oracle 11g como sistema gestor de bases de datos para este proyecto ya que satisface las necesidades.

En cuanto al sistema operativo (S.O.) de la máquina servidora, se utilizará un sistema operativo un poco más potente que el de la máquina cliente y deberá ser un S.O. específico para servidores, ya que estos ofrecen una mejora notable en lo que a comunicaciones y configuraciones se refiere. Unas posibles alternativas actuales serían utilizar Windows Server (2003 o 2008) o Alguna distribución Linux como Ubuntu Server.

Con respecto al hardware del servidor ha de ser bastante más potente que el de las máquinas clientes, ya que es el que se va a encargar de recibir la información concurrente por parte de estas. Por lo cual sería recomendable disponer de una máquina que disponga de más de un procesador, para así poder hacer frente a cualquier número de peticiones de servicio que reciba.

En cuanto a memoria RAM se refiere, sería recomendable disponer de la cantidad que actualmente ofrecen los servidores en el mercado, pero con un margen considerable, por tanto, esta cantidad podría oscilar en torno a 4 – 16 Gb de memoria RAM.

3.2.2 Especificaciones técnicas

Como se ha especificado en el apartado anterior, ningún requisito respecto a tecnología es inviable, y todos ofrecen un rango bastante amplio de decisión en cuanto a potencia, rendimiento y presupuesto. En este apartado se detallarán las decisiones tomadas en cuanto a especificaciones técnicas para comenzar con la realización del sistema.

3.2.2.1 Cliente

Como se ha comentado, cualquier ordenador comercial cumple con creces los requisitos técnicos para la parte del cliente, se exponen los requisitos mínimos orientativos.

Configuración mínima:

- Sistema Operativo: Microsoft Windows XP Professional.
- Procesador: AMD Athlon 64, 2000 MHz (10 x 200) 3000+, ó correspondiente Intel.
- Memoria RAM: 512 MB.

Almacenamiento: Disk Device (80 GB, 7200 RPM, SATA).

3.2.2.2 Servidor

La máquina servidora debe de disponer de una configuración típica para ordenadores que desempeñen este tipo de labores, son especificaciones más restrictivas que la máquina cliente.

Se propondrán dos configuraciones, mínima y recomendada, ambas completamente viables actualmente ya que se corresponden con las dos máquinas que el grupo LABDA pone a disposición de este proyecto. La configuración mínima pertenece a un equipo con más de cinco años de antigüedad y la configuración recomendada pertenece a un servidor con unos tres años de antigüedad, por tanto cualquier servidor comercial actual podrá satisfacer los requisitos técnicos.

Configuración mínima:

- Sistema Operativo: Microsoft Windows Server 2003, Enterprise Edition.
- Procesador: 2x Intel Pentium IIIE, 1000 MHz (7.5 x 133).
- Memoria RAM: 1024 MB.
- Almacenamiento: SCSI Disk Device (36 GB, 10000 RPM, Ultra320 SCSI).

Configuración recomendada:

- Sistema Operativo: Microsoft Windows Server 2003, Enterprise Edition. o superior.
- Procesador: AMD Opteron Quad-Core Processors: 2356 (2.3 GHz, 75W ACP), 2354 (2.2 GHz, 75W ACP), 2384 (2.7 GHz, 75W ACP), 2376 (2.3 GHz, 75W ACP).
- Memoria RAM: 4096 MB.

Almacenamiento: 146 G 15K rpm 3.5" SAS drive.

3.3 Pruebas definidas sobre los requisitos

En este apartado se detallaran las pruebas de validación necesarias para comprobar que el sistema cumple satisfactoriamente los requisitos específicos definidos en el apartado 3.2.

Cada prueba atenderá a la validez de un requisito en concreto, pero es posible que una prueba valide más de uno, esto ocurre ya que algunos requisitos tienen referencias directas con otros, lo cual se podrá comprobar en la matriz de correspondencia requisito-prueba que se incluye tras la definición de las rpuebas.

Prueba 1: Seleccionar un corpus y un escenario contenido en el mismo. Crear una nueva muestra vacía, comprobar que se activa la fase de pre-segmentación con un diálogo en blanco editable por el usuario.

Prueba 2: Seleccionar una muestra que se haya analizado previamente. Verificar que se activan todas las fases de análisis que ya han sido superadas con la información editada previamente en cada una de ellas.

Prueba 3: Seleccionar un corpus y comprobar que es posible modificar su descripción. Realizar la misma operación al seleccionar una muestra.

Prueba 4: Realizar un análisis total o parcial de una muestra y comprobar que se guarda correctamente. Para ello, tras guardarlo, verificar que se puede recuperar de nuevo y que no se ha perdido ningún detalle del análisis realizado.

Prueba 5: Verificar la existencia de una franja de texto informativo que cambia cada vez que se realiza una modificación en cualquiera de las fases.

Prueba 6: Provocar un error para comprobar que el sistema advierte al usuario de la situación mediante una ventana de tipo “pop-up”.

3.3.1.1 Pruebas para la fase pre-segmentación.

Prueba 7: Comprobar que en la fase de pre-segmentación existe una opción que permite obtener el texto de una muestra a través de un fichero externo. Verificar que en cualquier momento de la pre-segmentación el diálogo que se está analizando podrá sustituirse por otro almacenado en un fichero del disco duro.

Prueba 8: Verificar que al seleccionar un fragmento del diálogo y asignarle un rol concreto, se cambia el color de la fuente de dicho fragmento.

Prueba 9: Confirmar que al dejar fragmentos de texto sin rol asignado, el chequeo advierte al usuario de la situación y no le permite acceder a las siguientes fases del análisis. Verificar que el chequeo de una muestra vacía devuelve el mismo error.

3.3.1.2 Pruebas para la fase microanálisis.

Prueba 10: Comprobar que es posible la asignación de un conjunto de actos comunicativos a cada intervención tanto manual como automáticamente.

Prueba 11: Verificar que en esta fase existe un enlace con las aplicaciones *Cognos.CA* y *Cognos.NL*.

Prueba 12: Confirmar la posibilidad de asignar variables de contexto a una intervención manualmente. Además constatar que dichas variables son añadidas automáticamente cuando se ha realizado un análisis automático.

Prueba 13: Probar que el chequeo del microanálisis es correcto siempre y cuando a todas las intervenciones se les haya asignado al menos un acto comunicativo.

3.3.1.3 Pruebas para la fase realización temporal.

Prueba 14: Comprobar que en la fase de realización temporal se visualizan todas las intervenciones del diálogo siguiendo una línea temporal que representa el instante en el que cada una de ellas tuvo lugar. Para ello, verificar que cada intervención aparece en una línea diferente y que su comienzo es inmediatamente posterior al de la intervención previa.

Prueba 15: Corroborar que el color de cada intervención corresponde con el color del rol asignado en la fase de pre-segmentación de la muestra.

Prueba 16: Asegurarse de que es posible añadir un silencio delante o detrás de una intervención de la muestra eligiendo la duración del mismo. Constatar que dicho silencio es identificable visualmente por el usuario y revisar la posibilidad de modificar su duración.

Prueba 17: Confirmar la posibilidad de asignarle un tipo a cada silencio introducido en la muestra y que dichos tipos son: intervalo, lapso, pausa o pausa oralizada.

Prueba 18: Constatar que se pueden introducir solapamientos entre intervenciones de diferente rol y que es posible decidir el número de palabras que se solapan. Verificar que los solapamientos son fácilmente identificables gráficamente y que se puede modificar el número de palabras solapadas pudiendo incluso ser un solapamiento de tipo no físico en el que no se solapa ninguna palabra.

Prueba 19: Observar que es posible definir el cambio de tono que ha realizado el hablante de la intervención solapada, el cual puede disminuir, aumentar, mantenerse o abortar. Asimismo, advertir que se puede modificar el tipo de cambio de tono de la intervención que solapa el cual puede aumentar, disminuir, mantenerse o robar el turno de la intervención anterior.

Prueba 20: Comprobar que cuando existe un silencio entre dos intervenciones y se inserta un solapamiento entre las mismas, el silencio desaparece, sea cual sea su duración y aparece un solapamiento no físico.

Prueba 21: Verificar que en caso de existir un solapamiento entre dos intervenciones, al introducir un silencio entre las mismas, desaparece el solapamiento independientemente del número de palabras involucradas en el mismo y se crea un silencio por defecto de medio segundo de duración.

Prueba 22: Corroborar que existe la opción de reproducir un archivo de audio desde la aplicación para facilitar la identificación de las pausas y solapamientos de la muestra cargada.

3.3.1.4 Pruebas para la fase segmentación.

Prueba 23: Comprobar que en esta fase es posible crear un segmento a partir de un determinado conjunto de intervenciones (o actos comunicativos) elegidos por el usuario.

Prueba 24: Demostrar que a cada intervención (o conjunto de actos comunicativos asociados) se le puede asignar una secuencia de tipo: presecuencia, apertura, desarrollo, oscurecimiento o abandono, reapertura, cancelación o cierre.

Prueba 25: Verificar que se visualiza globalmente la estructura de segmentos creados junto con las intervenciones involucradas en cada uno de ellos. Comprobar que además se visualiza el tipo de secuencia que tiene asignada cada intervención.

Prueba 26: Demostrar que es posible crear un segmento a partir de un conjunto de intervenciones del diálogo. Comprobar que se puede modificar el conjunto de intervenciones asociadas en un principio a dicho segmento. Probar que es posible desplazar el segmento a la izquierda y a la derecha dentro de la tabla de segmentos y, por último, corroborar que dicho segmento puede ser eliminado.

Prueba 27: Corroborar que existe un segmento base por defecto que abarca todo el diálogo y sobre el cual no está permitida ninguna operación que lo modifique.

Prueba 28: Comprobar que se puede modificar el padre de un segmento, pudiendo subir en la jerarquía de segmentos (su padre pasa a ser su hermano) o bajar dentro de la misma (su hermano pasa a ser su padre).

Prueba 29: Probar que a un determinado segmento se le puede asignar una intención definida por un identificador y unas características. Verificar que es posible cambiar tanto el identificador como las características de una intención.

Prueba 30: Comprobar que está permitido asignar el orden de ejecución entre dos segmentos (antecedente y consecuente) eligiendo entre los tipos: inicio-inicio, inicio-fin y fin-fin. Verificar que no es posible crear precedencias recíprocas.

Prueba 31: Demostrar que a un segmento se le puede asignar un conjunto de presuposiciones que son definidas por un nombre y un valor.

Prueba 32: Justificar la posibilidad de incluir los cambios de foco producidos en el diálogo pudiendo introducir saltos regulares (por defecto) o excepcionales en las presecuencias, secuencias de apertura y reaperturas.

3.3.2 Matriz de trazabilidad

En la Tabla 1 se muestra la correspondencia que tienen las pruebas con respecto a los requisitos específicos definidos anteriormente. Cada requisito deberá estar validado como mínimo por una de las pruebas detalladas en el apartado 3.3.1. Por otro lado, es posible que una prueba valide más de un requisito.

La totalidad de las pruebas han sido definidas exclusivamente sobre los requisitos específicos. La matriz de trazabilidad no mostrará los requisitos generales del sistema puesto que las pruebas a realizar para la validación de los mismos son generales para la ejecución de cualquier aplicación software.

Requisito-	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
Prueba	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
R14	X																															
R15	X																															
R16	X																															
R17		X																														
R18			X																													
R19				X																												
R20					X																											
R21						X																										
R22							X																									
R23								X																								
R24									X																							
R25										X																						
R26											X																					
R27												X																				
R28													X																			
R29														X																		
R30															X																	
R31																X																
R32																	X															
R33																		X														
R34																			X													
R35																				X												
R36																					X											
R37																						X										
R38																							X									
R39																								X								
R40																									X							
R41																										X						
R42																											X					
R43																												X				
R44																													X			
R45																														X		
R46																															X	
R47																																X

Tabla 1: Matriz de trazabilidad

3.4 Metodología de desarrollo

Como la mayoría de proyectos de investigación de esta magnitud, el ciclo de vida que sigue es el modelo en espiral, es un modelo que combina las características de los modelos en cascada y de prototipos. El ciclo de vida en espiral está pensado para proyectos largos, caros y complicados en el cual la línea base transcurre a través de sus etapas varias veces a lo largo de su vida.

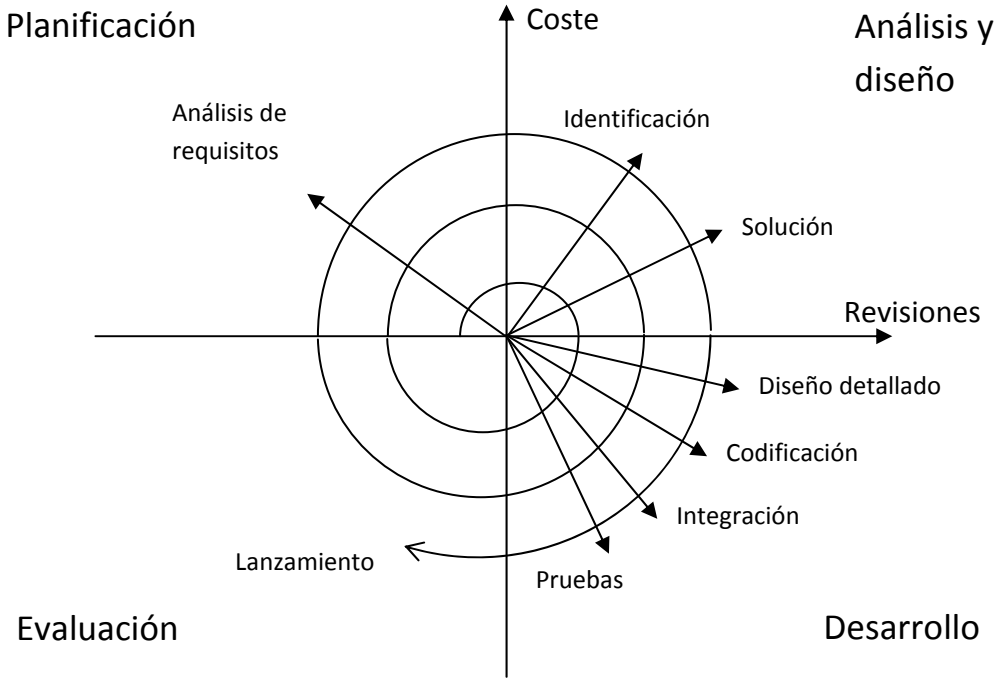

Ilustración 3.1: Ciclo de vida en espiral

Planificación: Fase en la cual se determinan los objetivos del proyecto, se identifican los requisitos. Esta definición de requisitos se irá enriqueciendo cada vez que la línea base transcurra a través de esta fase añadiendo, eliminando o modificando requisitos como a su vez modificando su prioridad.

Análisis de riesgos: En esta fase se identifican y se resuelven los riesgos mediante un estudio de viabilidad de la iteración. El producto esta fase por cada iteración es un prototipo viable con la menor cantidad de riesgos posible.

Desarrollo: Es en esta fase en la cual se realizan la codificación y pruebas de cada prototipo, dando como resultado un producto por iteración. A lo largo de esta fase se pueden englobar tareas tales como la realización de un diseño detallado o la integración del sistema.

Evaluación: En esta fase se evalúa el producto de la fase anterior, evaluación en la cual se extraen las deficiencias de este, las cuales son empleadas para realizar la planificación de la siguiente fase que será de nuevo la planificación hasta que el sistema llegue al punto de lanzamiento.

Al tratarse de un proyecto de investigación, se hace difícil determinar cuántos ciclos del ciclo de vida en espiral se realizarán hasta tener el producto final ya que los requisitos se irán modificando y ampliando a lo largo del proceso de desarrollo. En concreto, en el primer ciclo se comenzará por la planificación general que conducirá a la elección de tecnologías involucradas para su viabilidad y a la construcción de una maqueta (no funcional) que constituirá en esqueleto de la aplicación final. En el segundo ciclo se introducirán parte de los requisitos que definen las funciones generales de la aplicación. En los siguientes cuatro ciclos se introducirán los requisitos correspondientes a la primera, segunda, tercera y cuarta fase del análisis. Para finalizar, en un último ciclo se refinarán los requisitos generales a las cuatro fases.

4 Análisis y diseño

Una vez realizado un estudio en profundidad de los requisitos generales y específicos definidos sobre la aplicación, en las siguientes secciones de este apartado se procederá a realizar el análisis y diseño del sistema. El fin que se persigue es encontrar una solución a alto nivel (y sin entrar en detalles de implementación) que satisfaga las necesidades que plantea el sistema.

En este apartado se definirá en primer lugar, la arquitectura del sistema, en segundo lugar el diseño arquitectónico de la aplicación a nivel conceptual, seguidamente se realizará un estudio de las necesidades de la información con la que opera el sistema y, por último, el diseño de las interfaces del sistema con las que interactuará el usuario.

4.1 Arquitectura física del sistema

Como se ha mencionado, el sistema se basa en una arquitectura cliente-servidor en la cual las aplicaciones clientes realizan la anotación de un corpus lingüístico y el conocimiento se almacena en una base de datos común presente en el servidor. El servidor permanecerá activado permanentemente para que siempre que un cliente necesite realizar anotaciones, disponga tanto del conocimiento previo como la posibilidad de insertar nuevo, sin necesidad de realizar un arranque de dos máquinas.

Una arquitectura de este tipo mejora considerablemente la velocidad de adquisición de conocimiento ya que múltiples usuarios pueden anotar un mismo corpus en paralelo. Otra posibilidad sería utilizar este paralelismo para verificar y validar dos anotaciones del mismo corpus y compararlas, para, de este modo, obtener un conocimiento más pulido.

En la Ilustración 4.1 se aprecian los tres servidores, que son los que poseen, respectivamente, la base de datos Cognos, la base de conocimiento para el modelo de diálogo y la base de datos para la futura herramienta Cognos.Dial Global. Por otro lado, se muestran las diferentes aplicaciones que actúan como cliente, realizando inserciones y consultas en estas bases de datos.

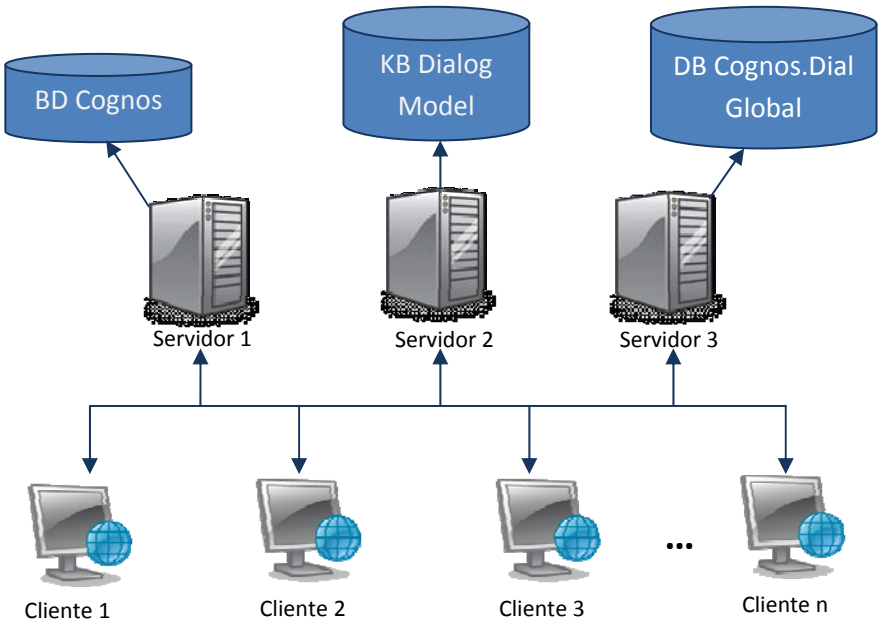

Ilustración 4.1: arquitectura cliente-servidor

Para que esto sea posible, se requiere que todas las máquinas que ejecutan el cliente, dispongan de conexión a internet y acceso a la máquina servidora

4.2 Arquitectura funcional del sistema

Una vez definida la arquitectura del sistema, en este apartado se explicará el diseño arquitectónico que se ha elaborado con la intención de facilitar la posterior implementación de la aplicación. Se ha decidido desarrollar un diagrama de componentes en UML como medio visual para representar el diseño del sistema a alto nivel. Este diagrama permite representar cómodamente cómo un sistema software es dividido en componentes y además muestra las dependencias entre estos componentes.

A la hora de planear cual sería el diseño del diagrama de componentes, se ha tomado como referencia el patrón de arquitectura software Modelo-Vista-Controlador (MVC), el cual separa los datos de la aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Dicha decisión de diseño ha sido tomada puesto que este patrón encaja en gran medida con las necesidades funcionales que presenta el sistema al desacoplar la lógica de la interfaz de usuario (vista) de los almacenes de datos (modelo) y de la lógica de negocio (controlador). La lógica de un interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio, por lo tanto, con el patrón MVC, las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

En la Ilustración 4.2 se puede observar el diagrama de componentes desarrollado, en el cual, de acuerdo con el patrón MVC, aparecen tres paquetes principales denominados *Model* (modelo), *View* (vista) y *Controller* (controlador). Al mismo

tiempo, se pueden visualizar las relaciones existentes entre los tres paquetes. El paquete *View* depende funcionalmente de los paquetes *Controller* y *Model*, ya que desde la interfaz de usuario es necesario acceder tanto a la lógica del negocio como a los almacenes de datos. Por otra parte, el paquete *Controller* debe tener acceso al lugar de almacenamiento de los datos, de ahí la relación representada con el paquete *Model*.

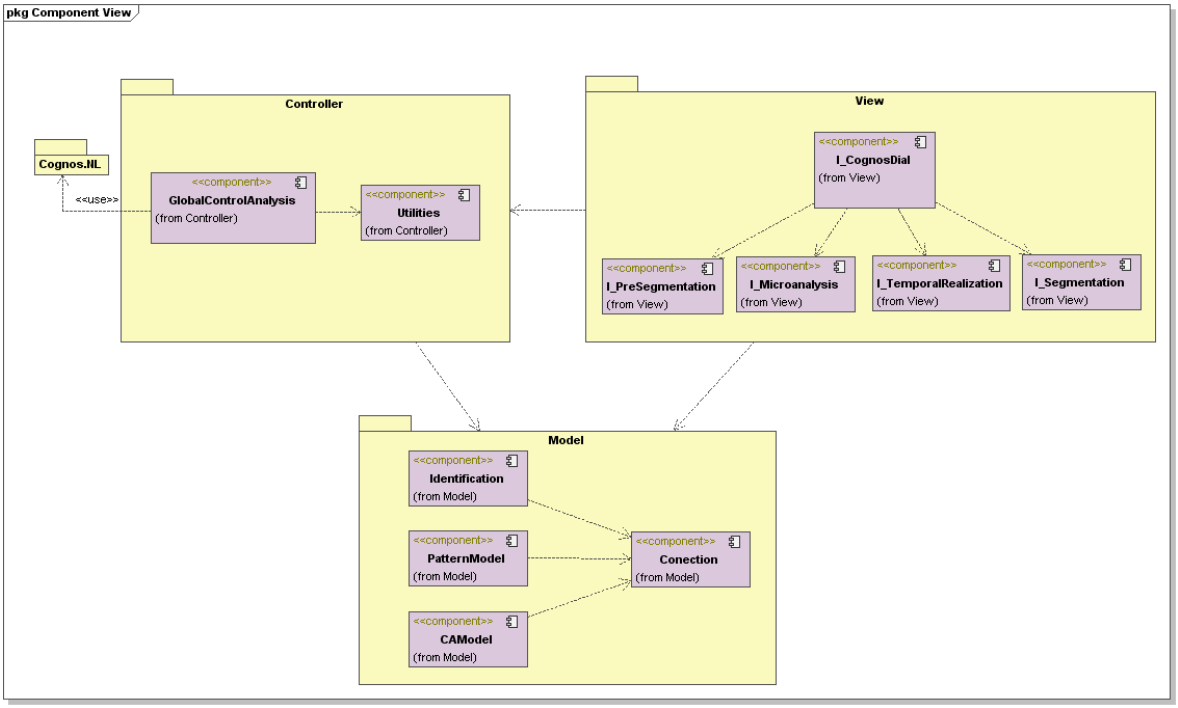

Ilustración 4.2: Diagrama de componentes

En los siguientes sub-apartados se definirá la funcionalidad que recoge cada uno de los paquetes del diagrama de componentes. Asimismo, se realizará tanto la identificación como la descripción general de los componentes mostrados en el interior de dichos paquetes.

4.2.1 Modelo

El Modelo será el responsable de acceder a la capa de almacenamiento de datos, en este caso, el acceso se realizará a un sistema gestor de base de datos. Por consiguiente, el Modelo deberá proporcionar un mecanismo de conexión con el SGBD además de dar soporte a las operaciones de inserción, borrado y actualización de los datos almacenados en el mismo. Además este paquete se encargará de realizar el encapsulamiento de los datos almacenados en objetos accesibles por los otros dos paquetes del sistema: Controlador y Vista.

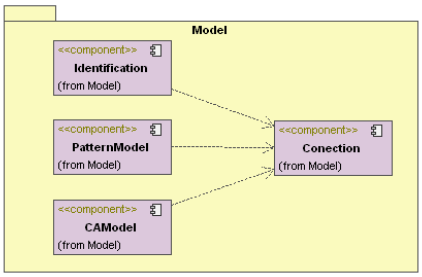

Ilustración 4.3: Modelo

Como se puede apreciar en la Ilustración 4.3, dentro del paquete Modelo existen cuatro componentes: *Conection, Identification, PatternModel y CAModel.* A continuación se describirá cual es el objetivo y las principales funciones de cada uno de ellos.

4.2.1.1 Model.Conection

El componente *Conection* representará la interfaz de comunicaciones entre el SGBD y los demás componentes del paquete Modelo*: Identification, PatternModel y CAModel*. Su principal objetivo será gestionar la conexión con el SGBD en el cual se encontrarán almacenados los datos y dar el soporte necesario para el tratamiento de sentencias PL/SQL de inserción, borrado y actualización. Del mismo modo, hará posible la ejecución de consultas en el mismo lenguaje sobre los datos almacenados.

4.2.1.2 Model.Identification

Este componente permitirá la ejecución de las operaciones específicas de inserción, actualización y recuperación de información que requiere el sistema sobre los datos almacenados de corpus, escenarios, muestras, intervenciones y frases. El componente *Identification* dependerá de *Conection* para lograr comunicarse y operar con el SGBD.

4.2.1.3 Model.PatternModel

Este componente ha sido desarrollado para el proyecto *Cognos.NL* y por lo tanto es caja negra para este proyecto. No obstante, su función es similar a la del componente *Identification,* con la distinción de que la manipulación de datos se encuentra definida para elementos propios del sistema *Cognos.NL* necesarios para inferir el conocimiento que se utiliza en la fase de microanálisis.

4.2.1.4 Model.CAModel

Análogo a los dos anteriores. Se encargará de ofrecer el acceso a todos los datos referentes a los actos comunicativos que el controlador y la vista necesiten. Las inserciones de este tipo de información las realiza la herramienta *Cognos.CA,* pero la obtiene COGNOS.Dial para permitir ofrecer la selección de actos comunicativos genéricos. Por tanto este componente ofrecerá funcionalidad de acceso y obtención aunque únicamente esta última sea requerida para este proyecto.

4.2.2 Vista

El paquete Vista (Ilustración 4.4) será el responsable de mostrar, mediante el uso de interfaces gráficas, la información almacenada en el modelo, además será el encargado de recibir información del usuario mediante acciones efectuadas por el mismo a través de dichas interfaces.

A través de las interfaces de usuario incluidas en la Vista será posible introducir toda la información resultante de la anotación de corpus para las cuatro fases en las que se centra el proyecto. Por este motivo, el componente Vista es de especial importancia y en él se recogerá una gran carga de trabajo, ya que se precisan numerosos objetos gráficos para representar tanto la información como las acciones que deben encontrarse disponibles en cada una de las interfaces de usuario de las que constará la aplicación.

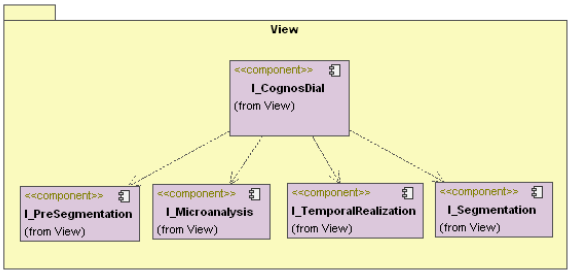

Ilustración 4.4: Vista

4.2.2.1 View.I_CognosDial

I_CognosDial es el principal componente del paquete Vista y representará la interfaz de usuario global del sistema. En él se recogerán las interfaces gráficas que dan acceso a los controles globales de las cuatro fases del proyecto.

Este componente gráfico contendrá una barra de herramientas desde la cual será posible la ejecución de las operaciones globales a las cuatro fases del análisis. Desde la misma se permitirá el acceso a las siguientes operaciones:

- **Selección de corpus, escenarios y muestras:** Aparecerán tres cajas de selección que contienen la totalidad de los corpus, muestras y escenarios que se encuentran almacenados en la base de datos. Mediante dichos selectores será posible elegir el corpus, el escenario y la muestra sobre los cuales se desea realizar alguna operación.
- **Creación de nuevos corpus, escenarios y muestras:** La barra de herramientas contendrá un botón desde el cual será posible crear nuevos corpus. Para crear un escenario, será obligatorio que el usuario elija previamente el corpus al que pertenece a través del selector de corpus. De la misma manera, para crear una muestra será necesario elegir el corpus y el escenario a los cuales pertenece.

- **Recuperación de una muestra:** Es necesaria la existencia de un botón que recupere los datos de una muestra de la base de datos. Previamente, el usuario ha de elegir a través de los selectores la muestra que desea recuperar.
- **Borrado de una muestra:** Será posible realizar el borrado de muestras almacenadas en la base de datos a través de uno de los botones existentes en la barra de herramientas.
- **Cerrado de una muestra:** Un botón de la barra de herramientas permitirá cerrar la muestra sobre la que se está trabajando, avisando al usuario mediante una ventana de tipo pop-up de la posible pérdida de información.
- **Visualización y modificación de descripciones:** Mediante un botón de la barra de herramientas se accederá a la descripción de los corpus y los escenarios. Previamente a la pulsación de dicho botón, ha de seleccionarse el corpus correspondiente y de esta manera se visualizará una interfaz que recoge la descripción del corpus y permite modificarla. Para acceder a la descripción de un escenario, es necesario seleccionar anteriormente tanto el corpus como el escenario correspondiente y en una interfaz análoga a la descripción aparecerán las descripciones de ambos.
- **Chequeo:** La operación de chequeo necesaria en tres de las fases del análisis (pre-segmentación, microanálisis y segmentación) será accesible mediante un botón dentro de la barra de herramientas.
- **Exportación e importación:** La barra de herramientas también posibilitará realizar tanto la exportación como la importación del análisis realizado sobre una muestra. Para dichas operaciones se creará o se leerá, respectivamente, de un fichero XML.
- **Upload de un análisis:** Botón que realizará la operación de subida a la base de datos del análisis completo realizado sobre la muestra.
- **Ayuda:** Botón dentro de la barra de herramientas que dará acceso a una ventana en la cual se muestra información de ayuda sobre el manejo de la aplicación.
- **Cerrar la aplicación:** Por último, la barra de herramientas contendrá un botón para cerrar la aplicación.

Este componente contiene las interfaces necesarias para mostrar al usuario información relativa al estado del análisis, así como de advertirle de cualquier situación de error (mediante ventanas emergentes) en caso de que existan inconsistencias durante el proceso de anotación.

Desde la vista I_CognosDial será posible acceder a las interfaces gráficas concretas de cada fase de la anotación. Dichas interfaces se recogen en los restantes componentes del paquete, correspondiendo cada uno de ellos a una fase del análisis de corpus y serán modeladas mediante pestañas contenidas en la interfaz global I_CognosDial.

4.2.2.2 View.I_PreSegmentation

El componente I_PreSegmentation contendrá una interfaz gráfica adaptada para efectuar la primera etapa de la anotación de una muestra. Esta interfaz será accesible a modo de pestaña desde la interfaz global I_CognosDial.

Dentro de la pestaña aparecerá un área de texto donde se visualizará el diálogo de la muestra y sobre la cual el usuario podrá seleccionar porciones del mismo y asignarle un rol determinado. Para permitir la asignación de un rol existirá una caja de selección que contendrá los nombres de los dos roles de la muestra que se está analizando. Cuando se le asigne a un fragmento de diálogo un rol determinado, dicho fragmento de texto se mostrará del color estipulado para ese rol.

Por último, dentro de la pestaña de la fase pre-segmentación se incorporará un botón desde el cual se tendrá acceso a una ventana que permitirá la carga de un diálogo desde un fichero almacenado en cualquier directorio de la máquina.

4.2.2.3 View.I_TemporalRealization

Este componente contiene una interfaz gráfica adaptada para permitir ejecutar las operaciones que satisfacen los requisitos específicos de la fase de realización temporal. Al igual que el componente I_PreSegmentation, esta vista será accesible a modo de pestaña desde la vista global I_CognosDial.

Esta interfaz debe contener un panel donde se visualiza el diálogo ya dividido en expresiones. Debe representarse la línea temporal que siguen dichas expresiones; para ello, cada expresión se mostrará en una línea diferente dentro del panel y comenzará en el punto en el que acabe la anterior. El objetivo que persigue esta representación es facilitar al usuario la introducción de silencios y de solapamientos entre expresiones.

Desde esta vista, será posible realizar las siguientes operaciones:

- **Introducir un silencio entre dos expresiones:** La interfaz incluirá un botón para mover a la derecha una expresión. Al seleccionar una expresión y pulsar el botón, la expresión se moverá hacia la derecha en el panel de expresiones y aparecerá una marca de silencio de tipo genérico entre la línea de discurso seleccionada y la anterior.
- **Modificar la duración de un silencio:** La duración de un silencio ha de visualizarse gráficamente en el panel de expresiones. La duración mínima de un silencio está estipulada en 0,5 segundos y a partir de ahí podrá aumentarse mediante un selector de duración que contiene valores fijos

desde 0,5 segundos en adelante y con un tamaño de salto también de 0,5. Otra vía posible para modificar este parámetro consiste en pulsar el botón mover derecha repetidas veces. Cada vez que se presione se aumentará la duración en 0,5 segundos y, al mismo tiempo, la distancia dentro del panel entre las expresiones involucradas aumentará dependiendo de los segundos que dure el silencio.

- **Modificar el tipo de un silencio:** Un silencio puede ser de tipo intervalo, lapso, pausa o pausa oralizada. Para asignarle un tipo a un silencio a través de la interfaz, deben aparecer cuatro botones de opción, cada uno con un nombre de tipo de silencio. La selección de uno de los botones de opción debe ser excluyente, de modo que un silencio sólo puede ser de uno de los cuatro tipos. Cada tipo de silencio tiene una representación propia que se indica en la Ilustración 4.5:

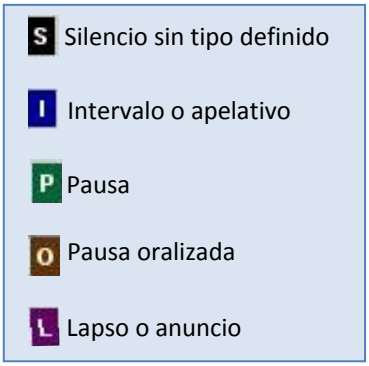

Ilustración 4.5: Representación de los tipos de silencio

- **Introducir un solapamiento físico entre dos expresiones:** Para permitir introducir un solapamiento físico entre dos expresiones, al igual que existe un botón de mover a la derecha una expresión, debe de aparecer uno análogo para moverla a la izquierda. Si el usuario selecciona una expresión y presiona dicho botón, la expresión se moverá una posición a la izquierda solapándose con la anterior en una palabra (unidad mínima de solapamiento). Hay que recordar que un solapamiento físico sólo es posible introducirlo cuando las expresiones involucradas sean de diferente rol.
- **Introducir un solapamiento no físico entre dos expresiones:** La interfaz contendrá una opción que permita determinar al usuario si existe un solapamiento no físico entre dos expresiones. Dicho solapamiento no tendrá ninguna restricción en cuanto a los roles involucrados en el mismo, pudiendo pertenecer ambas al mismo rol. Este tipo de solapamiento no será posible determinarlo con el botón que mueve a la izquierda una expresión, ya que en un solapamiento no físico no hay palabras que se solapen.

- **Modificar la cantidad de palabras solapadas:** La duración de un solapamiento o cantidad de palabras solapadas entre dos expresiones de diferente rol se determinará presionando varias veces el botón mover izquierda. Cada vez que se presione, se solapará una palabra más entre las dos expresiones involucradas y, al mismo tiempo, se apreciará dicho solapamiento en el panel de expresiones.
- **Modificar el tipo de cambio de tono:** Debe existir un control que permita modificar el cambio de tono que realizan los hablantes de dos expresiones que se solapan. Por tanto, existirá una caja de selección que permita definir el cambio de tono del hablante de la expresión solapada. Este selector contendrá los valores: disminuye, mantiene, eleva o aborta. Por la parte del hablante de la expresión que solapa, existirá otra caja de selección con los valores: baja, disminuye, mantiene, eleva o robo de turno.

Todas estas operaciones será posible realizarlas con respecto a la expresión anterior a la seleccionada o con respecto a la expresión siguiente. Por este motivo, la interfaz contendrá un panel con dos pestañas similares (expresión previa y expresión posterior) donde se recogerán todas las operaciones definidas en las viñetas anteriores.

Finalmente, para facilitar la identificación de silencios y solapamientos, esta interfaz contará con unos controles de audio básicos que permiten reproducir un fichero de audio almacenado en la máquina del usuario. Por tanto, dentro de la interfaz existirán tres botones: *abrir fichero de audio*, *reproducir/pausar* y *parar*.

4.2.2.4 View.I_Microanalysis

La interfaz correspondiente a la fase de microanálisis será accesible a través de la tercera pestaña de la vista global. Desde la vista I_Microanalysis será posible asignar uno o varios actos comunicativos a cada una de las expresiones del diálogo. La manera de modelarlo será incluyendo en la interfaz una lista con todas las expresiones de la muestra (manteniendo el color de los roles asignados en la fase de pre-segmentación) y a su derecha otra lista (con el mismo número de ítems en blanco) donde irá apareciendo el conjunto de actos comunicativos correspondiente a cada expresión.

Desde la interfaz I_Microanalysis será posible consultar los tipos de actos comunicativos disponibles para el corpus que contiene a la muestra que se está analizando y, para cada tipo, todas sus categorías con sus correspondientes valores asignados mediante la herramienta *Cognos.CA*. Para hacer posible esta operación, en una lista aparecerán todos los nombres de acto comunicativo disponibles para ese corpus y al seleccionar uno de ellos, se mostrarán tantas listas como categorías contenga dicho acto comunicativo. En la primera lista aparecerán todos los valores de la primera categoría y dependiendo del valor que se elija para ella, en la siguiente categoría aparecerán unos valores u otros. Lo mismo ocurre con las categorías sucesivas: al

seleccionar un valor para una, aparecen los valores válidos para la siguiente. Una vez que se hayan seleccionado los valores para cada categoría del acto comunicativo elegido, se mostrará en una lista que contiene los actos comunicativos concretos en modo textual siguiendo el siguiente formato:

<tipo CA>(<nombre categoría1> = <Valor seleccionado1>,…, <nombre categoría N> = <Valor seleccionado N>)

Los actos comunicativos concretos existentes en la lista, ya están preparados para ser asignados manualmente a una expresión. Esta acción se realizará presionando un botón de la interfaz. Por otro lado, los actos comunicativos concretos pueden reordenarse indicando así la prioridad (que indica el orden en el que suceden dentro de una expresión) mediante dos botones que suben o bajan un AC dentro de la lista.

Un modo alternativo de asignar los correspondientes actos comunicativos a las expresiones es realizar un análisis automático haciendo uso de un motor de inferencias integrado en la aplicación, cuyo desarrollo se debe a otro proyecto y que es incrustado como caja negra en este proyecto. El análisis automático podrá realizarse para la totalidad de las expresiones de la muestra o sólo para la expresión que seleccione en la lista el usuario.

Si el usuario no está de acuerdo con la asignación de actos comunicativos que ha realizado (él mismo o el análisis automático) para una expresión, será posible eliminar dicha asignación de AACC mediante un botón que ha de incluirse en la interfaz.

Además desde la vista I_Microanalysis el usuario podrá introducir variables de contexto con sus correspondientes valores y su relevancia, para lo que se necesitará una tabla de tres columnas, dos campos de texto editables para introducir tanto el nombre de la variable de contexto como su valor y tres botones de opción excluyentes para introducir su relevancia (0, 1 o n). Junto a la tabla de variables de contexto aparecerán tres botones cuya función será crear una nueva variable con su nombre, valor y relevancia, actualizar una de ellas o eliminarla de la tabla. En caso de que se elija realizar el análisis (de una expresión o de todas las existentes) automáticamente, la tabla de variables de contexto se rellenará automáticamente con los valores devueltos por el motor de inferencias.

Puesto que es probable que el usuario desee incluir un acto comunicativo que no se encuentra registrado en el corpus o incluso modificar uno existente, desde la vista I_Microanalysis será posible lanzar una instancia de la aplicación *Cognos.CA* (mediante un botón) para que introduzca o modifique los actos comunicativos que desee y así poder incluirlos en la fase de microanálisis.

Para finalizar, existirá también un botón que lance la aplicación *Cognos.NL* para permitir al usuario editar la gramática relajada de una expresión.

4.2.2.5 View.I_Segmentation

La última fase de anotación de corpus que forma parte del proyecto será accesible desde la cuarta pestaña de la vista general y quedará recogida dentro del componente vista I_Segmentation. En líneas generales, este componente vista ha de ofrecer controles gráficos para permitir anotar información acerca de segmentos, de secuencias y de gestión atencional que aparece en la muestra de diálogo.

En primer lugar, es necesario incluir una tabla que contenga todas las expresiones de la muestra, manteniendo en cada una de ellas el color de su rol (al igual que en las fases de realización temporal y microanálisis). Como cada expresión corresponde a uno o a varios actos comunicativos (que han sido editados en la fase de microanálisis), mediante un botón de la interfaz será posible cambiar (en cualquier momento del análisis de la segmentación) la visualización de la tabla de expresiones, mostrando en su lugar el AC o conjunto de AACC que corresponden con cada expresión. Por lo tanto, cuando se hable en las siguientes líneas de este apartado del término expresión hay que tener en cuenta que todas las operaciones definidas para la tabla de expresiones se hacen extensibles para los AACC que corresponden con dichas expresiones.

Esta interfaz debe permitir al usuario seleccionar conjuntos de expresiones y crear segmentos que las contenga. Para crear un nuevo segmento, es necesario que las expresiones puedan ser seleccionadas dentro de la tabla y mediante un botón se creará un nuevo segmento para ese conjunto de expresiones. Para mejorar la visualización e identificación de segmentos, para cada segmento se creará una nueva columna inmediatamente a la izquierda de la tabla de expresiones. En dicha columna aparecerá un corchete que engloba todas las expresiones contenidas dentro del segmento.

Al iniciar la fase de segmentación, aparecerá siempre el segmento base, el cual engloba todas las expresiones y sobre el que no será posible realizar ninguna operación de edición.

El orden de visualización de las columnas que representan los segmentos seguirá el orden en el que han sido creadas. Por lo tanto la columna situada más a la izquierda corresponderá al segmento base y la columna situada más a la derecha y lindante con el panel de expresiones corresponderá al último segmento creado.

Es indispensable tener una visión global de todos los segmentos junto con las expresiones contenidas en ellos de modo que se pueda identificar claramente cuáles son las expresiones contenidas dentro de un segmento. Las columnas creadas por cada nuevo segmento son seleccionables, de modo que al elegir una de ellas se seleccionarán automáticamente las expresiones que forman parte de ese segmento.

A continuación se indican las operaciones que pueden realizarse a nivel de edición gráfica de segmentos:

- **Crear un segmento:** Mediante un botón será posible crear una nueva columna (que corresponde a un segmento) inmediatamente a la izquierda de la tabla de expresiones. En dicha columna aparecerá, tal y como se ha definido anteriormente, un corchete que engloba las expresiones contenidas en el segmento.
- **Reasignar un segmento:** Es posible modificar el conjunto de expresiones que conforman un segmento. Para ello, la interfaz debe contener un botón para realizar dicha operación. Al seleccionar un segmento, si el usuario selecciona un conjunto de expresiones diferente al mostrado automáticamente y presiona el botón, se cambiará la estructura del segmento.
- **Eliminar un segmento:** Se incluirá un botón para eliminar un segmento creado previamente.
- **Mover a la derecha o a la izquierda un segmento:** Se requiere la existencia de dos botones: *mover derecha* y *mover izquierda.* Seleccionando un segmento y pulsando uno de ellos, será posible modificar el orden del segmento, moviéndolo una posición a la derecha o a la izquierda dentro del panel de columnas de segmentos. Este control gráfico permitirá solventar errores que puede cometer el usuario al realizar la segmentación.
- **Modificar el vínculo descomposicional:** Dentro de la interfaz, existirán una serie de controles que permitan modificar el segmento padre de un segmento seleccionado. De esta forma, mediante los botones *subir* y *bajar* será posible aumentar o disminuir un nivel a un segmento dentro de su vínculo descomposicional. Al presionar el botón *subir,* la estructura del segmento cambiará en la interfaz para englobar un conjunto diferente de expresiones. De la misma manera, al presionar el botón *bajar* y seleccionar su nuevo padre (entre sus segmentos hermanos) en una caja de selección, se modificará la estructura del segmento.

Hasta ahora, la única información que se ha editado para cada segmento es el conjunto de expresiones contenido en él, y las operaciones definidas anteriormente tienen que ver únicamente con las estructura del segmento. No obstante, cada segmento alberga más información susceptible de ser editada a través de la interfaz I_Segmentación. Por este motivo, deben incluirse los siguientes elementos:

- **Intención y descripción de un segmento:** Se requerirán dos campos de texto en los cuales introducir el nombre de la intención que identifica al segmento y su descripción. Dicha intención puede ser similar a otra previamente editada, por lo que será necesario incluir un botón que de acceso a una pequeña ventana en la cual sea posible elegir mediante un selector la

intención que desea ser reutilizada. En dicha ventana se podrá visualizar la descripción de la intención seleccionada (ya que así el usuario podrá identificar más fácilmente que el segmento que ha creado desarrolla la misma intención que el previamente editado).

- **Presuposiciones:** Cada segmento puede tener un conjunto de presuposiciones asociadas que deben de poder editarse a través de la interfaz de usuario. Cada presuposición queda definida por un nombre y un valor asociado, por lo que será necesario incluir una tabla con dos columnas (nombre y valor) junto con dos campos de texto para introducir ambos datos y un botón *añadir* para introducirlos en la tabla. De la misma forma que se pueden añadir presuposiciones para un segmento, también será posible descartarlas mediante un botón *eliminar*.
- **Vínculos de precedencia:** Dentro de una muestra de diálogo, es posible que un segmento preceda o sea consecuente de otro. Se hace necesario recoger esta información para realizar adecuadamente el análisis en la fase de segmentación. Por consiguiente, para editar los vínculos de precedencia de un segmento se necesitarán los siguientes objetos gráficos en la interfaz: tres botones de opción excluyentes que indican el tipo de precedencia (inicio-inicio, inicio-fin o fin-fin), un selector para elegir el segmento que está vinculado con el que está siendo editado y los botones *añadir precedente* y *añadir consecuente* donde se elige cuál es el segmento precedente y cuál es el consecuente en la relación de precedencia. Todos los vínculos de precedencia editados para un segmento podrán visualizarse a través de una tabla de tres columnas que indican, para cada relación de precedencia, el segmento precedente, el segmento consecuente y el tipo de precedencia.

Desde esta interfaz, también se editará la información relativa a la secuenciación. Para ello, a la derecha de la tabla de expresiones debe de aparecer una pequeña columna para visualizar el tipo de secuencia de cada expresión. Cada uno de los tipos se identificará por su inicial y un color de fondo como se muestra en la Ilustración 4.6.

Por defecto, al comenzar el análisis en esta fase, todas las expresiones estarán definidas como secuencias de desarrollo, no obstante, el usuario podrá cambiar el tipo de cada una de las secuencias al que considere más adecuado. Esta operación se realizará seleccionando la expresión en la tabla y eligiendo su nuevo tipo de secuencia, para lo cual es necesario que existan siete botones de opción excluyentes y cada uno de ellos con el nombre del tipo de secuencia.

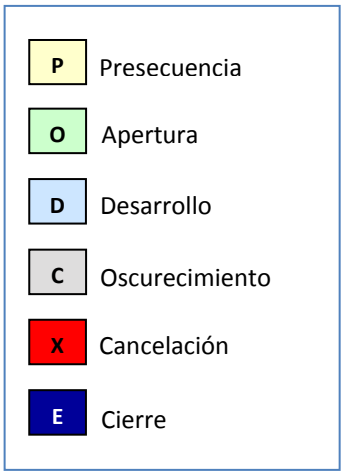

Ilustración 4.6: Representación de los tipos de secuencia

Por último, en la vista I_Segmentation será posible editar una pequeña parte de la gestión atencional, correspondiente a la identificación de saltos regulares y excepcionales que se pueden producirse en las secuencias de presecuencia, apertura y reapertura. La identificación de saltos regulares y excepcionales se realizará haciendo uso de dos botones de opción excluyentes. Por defecto, en estos tres tipos de secuencia, el salto estará definido como regular, sin embargo el usuario podrá identificar la existencia de un salto excepcional en una secuencia. Los saltos excepcionales, a nivel de interfaz, se señalarán en la tabla de expresiones, representando con un fondo de color naranja la expresión en la que se produce el salto.

4.2.3 Controlador

En el paquete Controller se definirán todas las reglas de negocio que permitirán ejecutar la totalidad de las operaciones que pueden ser ordenadas desde los objetos gráficos que se han definido previamente en componente interfaz. Para ello debe procesar la información que toma del paquete Vista para devolverle finalmente una respuesta, de modo que el usuario sea capaz de visualizar el resultado de la operación que ha ordenado.

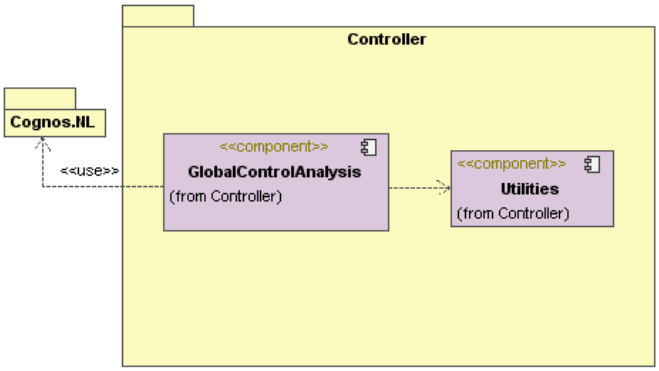

Ilustración 4.7: Controlador

Como se puede apreciar en la Ilustración 4.7: Controlador, dentro del paquete *Controller* existen dos componentes: *GlobalControlAnalysis* y *Utilities*. La función que realizan estos componentes será descrita en los siguientes subapartados. En la ilustración aparece además un paquete externo que se comunica con el componente *GlobalControlAnalisis*. Dicho paquete es externo al proyecto COGNOS.Dial, no obstante su propósito se explicará en el apartado 4.2.4.

4.2.3.1 Controller.GlobalControlAnalysis

El componente *GlobalControlAnalysis* recoge la mayor parte de la lógica de negocio de la aplicación. Dentro de él quedan definidas estructuras internas para el almacenamiento temporal de información relativa al análisis que se va desarrollando en la herramienta.

Tanto la información guardada temporalmente como las operaciones y reglas definidas sobre ella, corresponden a las cuatro fases del análisis. Para la fase de pre-segmentación se definirán todas las operaciones referentes a la asignación de roles a fragmentos de diálogo; para la fase de realización temporal se definirán las operaciones necesarias para incluir silencios y solapamientos entre expresiones; para el microanálisis toda la información y operaciones relativas a la asignación de actos comunicativos a expresiones y, por último, en la fase de segmentación se manejarán todos los datos de segmentos, secuencias y saltos junto con la definición de todos los procedimientos necesarios para ejecutar las operaciones que se encuentran definidas en la interfaz gráfica No obstante, la estructura de toda esta información junto con sus operaciones quedarán definidas más profundamente en el apartado de implementación 5.2.3.1.

4.2.3.2 Controller.Utilities

Este componente se encarga de ofrecer las funciones necesarias para realizar dos operaciones auxiliares. En primer lugar, ha de proporcionar las operaciones necesarias para permitir la reproducción de un fichero de audio (necesario para la fase de realización temporal) y en segundo lugar ha de dar soporte a la lectura de ficheros de texto externos (necesario para la fase de pre-segmentación).

4.2.4 NLP

Este paquete se encarga de inferir, sobre patrones almacenados en memoria principal, una expresión dada para así conseguir proponer un conjunto de actos comunicativos. Comprobará por tanto los patrones cargados buscando encajes en sus gramáticas y devolverá el conjunto de actos comunicativos asociados a dicho patrón.

Es un componente proporcionado y el propósito en este proyecto es investigarlo, comprenderlo y usar su tecnología para realizar las inferencias sobre el conocimiento almacenado en la base de datos.

En el apartado de implementación se realizará un estudio en detalle de sus estructuras y las equivalencias con los datos del sistema y de qué forma realiza sus operaciones.

4.2.5 JDBC

JDBC son las siglas de *Java Database Conectivity*, y es un API que permite la ejecución de operaciones de bases de datos desde el lenguaje de programación Java. Estas librerías son independientes del sistema operativo donde se ejecute o de la base de datos a la cual se accede. Se utilizan las sentencias en SQL que la base de datos admita.

Para utilizar una base de datos particular, se ejecuta la aplicación junto con la biblioteca de conexión apropiada al modelo de su base de datos (en este caso Oracle), y se accede a ella estableciendo una conexión. Para ello es necesario proveer el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí se puede realizar cualquier tipo de tareas con la base de datos a las que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

4.2.6 Javazoom

Javazoom es una biblioteca de libre distribución implementada en Java que permite decodificar y reproducir archivos de audio en formato MP3 (Javazoom). Esta librería es necesaria para reproducir el archivo de audio perteneciente al diálogo que desea ser analizado ya que proporciona las funciones necesarias para reproducir, pausar y parar este tipo de archivos.

4.3 Necesidades de información

En este apartado se analizará qué información requiere el sistema para poder ofrecer al usuario la funcionalidad requerida.

Antes de comenzar es necesario aclarar que la base de datos que muestra la Ilustración 4.8 forma parte de un conjunto de bases de datos que en su totalidad completarían la base de conocimiento COGNOS.

Ilustración 4.8: Base de conocimiento COGNOS.

Esta base de conocimiento la integra la base de datos del conocimiento obtenido del diálogo (poblada por Cognos.Dial), la base de datos de lenguaje natural (poblada por *Cognos.NL*), base de conocimiento de tareas, base de conocimiento del modelo de usuario, ontología de términos, base de conocimiento del modelo de situación, emoción y automodelo. De estas, únicamente las que figuran con letra de color negro disponen de soporte actualmente, pero en un futuro se dispondrá de una base de conocimiento totalmente integrada.

La Ilustración 4.9: Diagrama E/R de la BD, muestra el diseño en el modelo entidad/relación de la base de datos que la aplicación requiere, ya sea para consultar datos como para insertar información en ella.

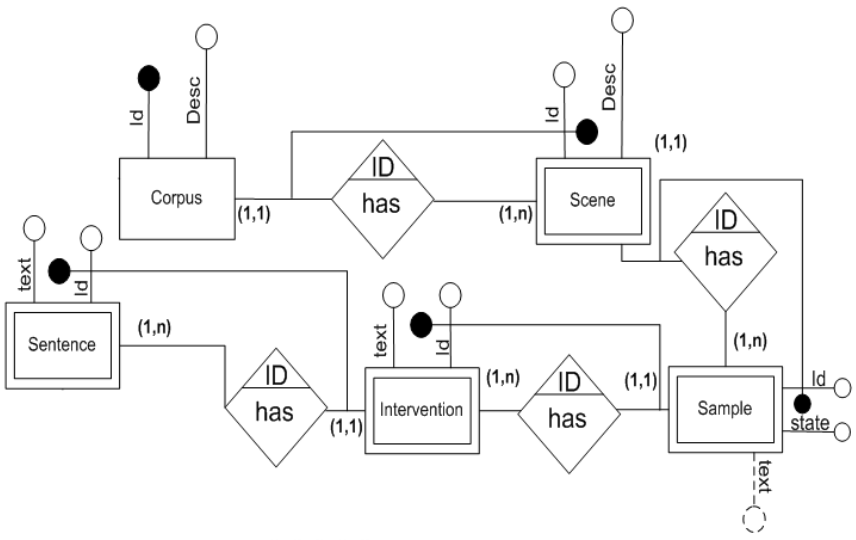

Ilustración 4.9: Diagrama E/R de la BD

Como se ha podido apreciar en el esquema, existen cinco entidades: Corpus, Scene, Sample, Intervention y Sentence y a continuación se definirá el propósito de cada una de ellas así como el de las relaciones existentes entre ellas.

Bajo una visión lógica, todo el conocimiento que se edita para cualquier modelo de conocimiento de este tipo, se engloba bajo dominios de interacción como ya se explicó en el apartado 2.1.2. Por consiguiente los escenarios pueden ser independientes del corpus y dependientes del dominio de interacción. Esta semántica lógica no está recogida en el diseño y por tanto la entidad Corpus realiza la función de dominio de interacción. En un futuro se mejorará el diseño de la identificación.

Tanto Corpus como Scene como Sample son entidades de identificación lógica, es decir, se dispone de una muestra de texto la cual ha de pertenecer irremediablemente a un escenario proveniente de un corpus para que dicha muestra aporte conocimiento. Si una muestra de diálogo no está identificada en un dominio de interacción, no constituye por sí sola información o conocimiento extraíble útil.

Por ello la lógica de este fragmento es que un Corpus puede contener varios escenarios y que un escenario puede contener varias muestras. Es en la entidad Muestra en la que se almacena el fragmento de diálogo al cual pertenecerán las intervenciones y cada intervención contendrá frases.

Se trata de cuatro relaciones débiles en identificación por lo comentado anteriormente, una intervención carece de información si no está asociada e identificada con un corpus concreto pero una expresión no tiene porqué. De forma similar ocurre lo mismo con el resto de entidades en este apartado de la base de datos con la excepción de la restricción lógica comentada en párrafos anteriores.

Se deja como línea futura ampliar la base de datos para que recoja la posibilidad de asignar análisis generales que no pertenezcan a ningún corpus en concreto. Actualmente para paliar esta carencia, se creará un corpus general perteneciente a la colección de corpus del sistema.

Además de la necesidad de la base de datos descrita cuyo principal propósito es permitir la identificación de corpus de diálogo, es necesario un mecanismo para almacenar el resultado de los análisis que han sido completados sobre cada muestra. Para ello, al finalizar las fases de anotación el resultado del análisis se recogerá, por cada muestra de diálogo en un fichero XML que sigue el formato definido en (D. Cuadra, 2009).

4.4 Diseño de la interacción

En este punto se realizará una presentación preliminar de las interfaces de usuario que contendrá la herramienta y se especificarán los objetos gráficos que son necesarios para realizar las cuatro fases de análisis de corpus. Se procurará realizar un diseño de interfaces cuidadoso de modo que se consiga una herramienta cómoda e intuitiva que mejore así la usabilidad de la herramienta por parte de usuarios poco familiarizados con el uso de ordenadores.

Dado que en este proyecto se van a implementar cuatro fases del análisis de corpus (pre-segmentación, realización temporal, microanálisis y segmentación), se ha decidido dividir la edición de cada una de las fases en pestañas. De este modo se facilita que en un futuro se introduzcan el resto de fases en la herramienta (compromiso, operativa y estructural).

Por otro lado, existen operaciones comunes que pueden ser realizadas en todas las fases (por ejemplo, el chequeo) y por este motivo se hace necesario incluir una barra de herramientas en la aplicación que sea accesible en cualquier punto del análisis.

A continuación se describirá más detalladamente el contenido de la barra de herramientas así como el de cada una de las pestañas, justificando las decisiones de diseño tomadas en cada caso.

4.4.1 Interfaz 1: Barra de herramientas global

Esta barra de herramientas aparecerá en la parte superior de la ventana principal de la aplicación y el usuario podrá arrastrarla a cualquier otra parte de la pantalla mediante la técnica *drag and drop*.

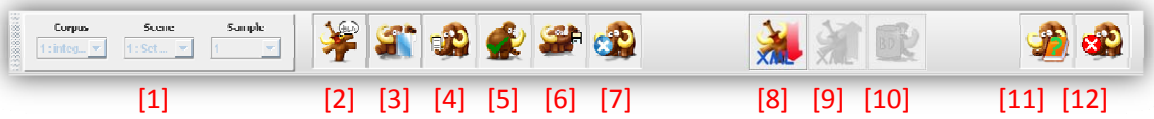

Ilustración 4.10: Barra de herramientas global

La barra contendrá tres selectores [1] para elegir corpus, escenarios y muestras y la siguiente serie de botones: descripción [2], abrir muestra [3], nueva muestra [4], chequear [5], guardar [6], cerrar muestra [7], importar fichero XML [8], exportar fichero XML [9], subir análisis a la BD [10], ayuda [11] y cerrar aplicación [12].

4.4.2 Interfaz 2: Pestaña de Pre-Segmentación

En la fase de realización temporal, el usuario puede asignar un determinado rol a un fragmento del diálogo. Para ello, en primer lugar es necesario mostrar el diálogo en un área de texto. En la Ilustración 4.11 y concretamente en [1] se muestra dicho área de texto. Como el diálogo puede ser muy extenso, se requiere que el área sea grande y por este motivo, ocupará prácticamente la totalidad de la pestaña.

A la hora de distinguir los roles, se ha creído conveniente asignar un color determinado a cada uno de modo que, en el momento que se asigna un rol a un fragmento de texto, el color de fuente cambia según el color del rol.

Para seleccionar el rol que se desea asignar, se incluirá un objeto de tipo *combobox* [2] debido a la comodidad que ofrece a la hora de seleccionar elementos. Aunque en esta versión de la herramienta únicamente se contemplarán dos roles, no se descarta que en un futuro se haga extensible a más.

Por último es necesario añadir un botón [3] que abra un cuadro de diálogo en el cual se pueda seleccionar un fichero de texto existente en cualquier directorio de la máquina.

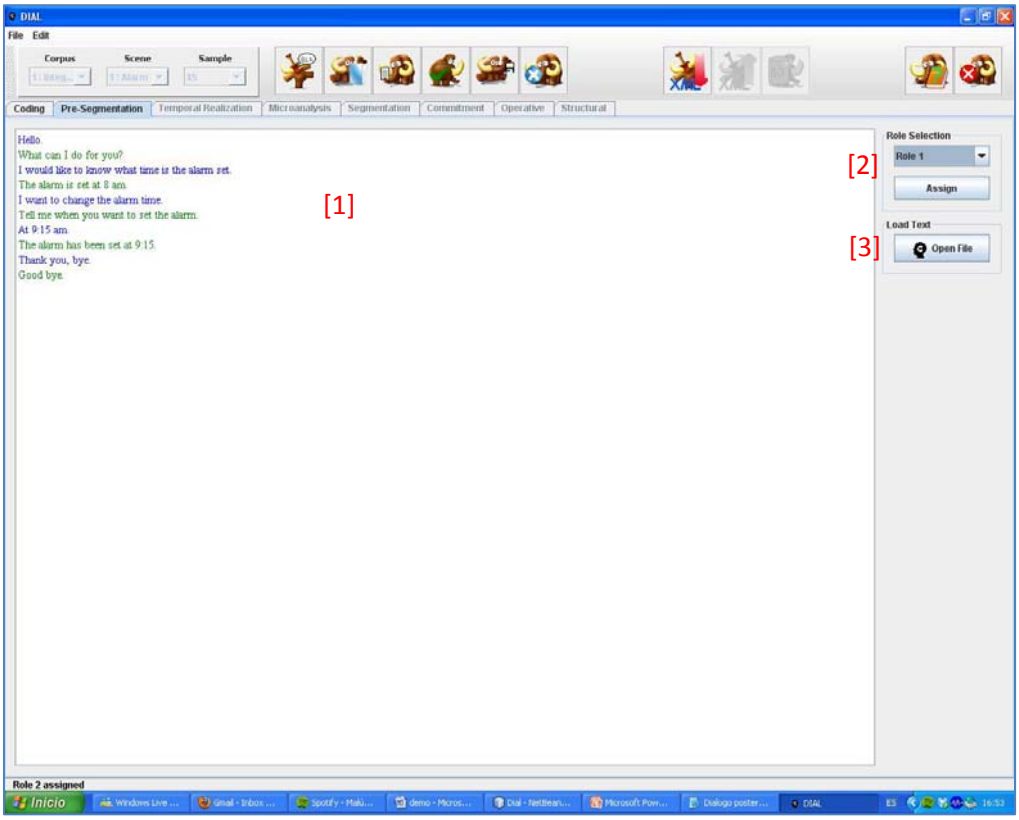

Ilustración 4.11: Pestaña de Pre-segmentación

4.4.3 Interfaz 3: Realización temporal

En esta pestaña se anotará la información de tipo prosódico que tiene lugar en el diálogo, en concreto las pausas y los solapamientos entre expresiones. Una manera intuitiva de visualizar gráficamente la existencia de una pausa o de un solapamiento entre dos expresiones es la que se muestra en la Ilustración 4.12, concretamente en [1]. Se puede apreciar la existencia de un panel que contiene todas las expresiones del diálogo, cada una de ellas mantiene el color del rol asignado en la etapa de pre-segmentación. Cada expresión se muestra en una línea diferente del panel, de modo que el usuario pueda mover cada expresión a la izquierda o a la derecha [3] para introducir un solapamiento con respecto a la intervención anterior o una pausa con respecto a la siguiente.

Para ayudar al usuario a reconocer la existencia de pausas y solapamientos, se han introducido unos controles, en concreto tres botones (reproducir/pausar, parar y abrir fichero de audio [2]) que aparecen en el panel derecho de la Ilustración 4.12 en la parte superior. Se situarán en ese lugar puesto que es una de las primeras acciones que el usuario realizará en la pestaña.

En el panel derecho de la Ilustración 4.12 se incluirá un panel de dos pestañas desde el cual se configurará el tipo de silencio [4] o de solapamiento [5] que se haya introducido con anterioridad o posterioridad a la expresión seleccionada en el panel. La existencia de dos pestañas permite una mayor flexibilidad a la hora de manejar tanto silencios como solapamientos, ya que se podrá hacer con respecto a la expresión anterior y con respecto a la expresión siguiente.

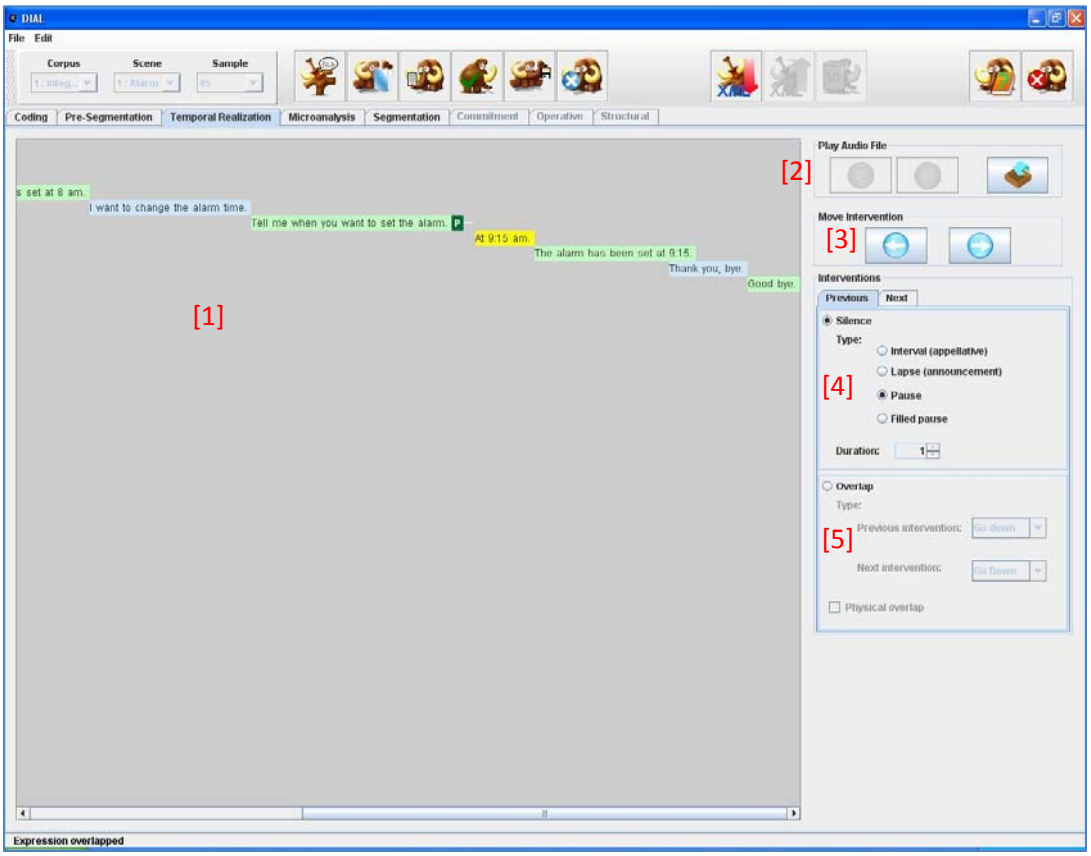

Ilustración 4.12: Pestaña de Realización Temporal

Como hay cuatro tipos de silencio posibles, el objeto gráfico que mejor se adapta en este caso son los botones de radio de selección exclusiva de modo que sólo se pueda seleccionar uno de los tipos. En el caso de la duración, al ser ésta numérica y de granularidad 0,5 segundos, el componente gráfico que mejor se adapta es el *jSpinner* que permite aumentar o disminuir la duración del silencio con variaciones fijas de medio segundo.

En el caso del solapamiento, se incluirán dos objetos de tipo *comboBox* con cuatro ítems cada uno que representan el tipo de cambio de tono de la expresión anterior y de la expresión siguiente. Por último, bastará un componente *checkBox* para identificar si el solapamiento es o no físico.

4.4.4 Interfaz 4: Microanálisis

En objetivo de esta fase es que el usuario asigne a cada expresión el conjunto de actos comunicativos que le corresponda. Por lo tanto, son necesarias dos listas amplias (ver Ilustración 4.13) de modo que en una de ellas se muestren las expresiones [1] y en la otra los actos comunicativos [2]. Para que se vea la correspondencia entre una expresión y su correspondiente conjunto de actos comunicativos, es necesario situar las dos listas con poca distancia de separación y a la misma altura.

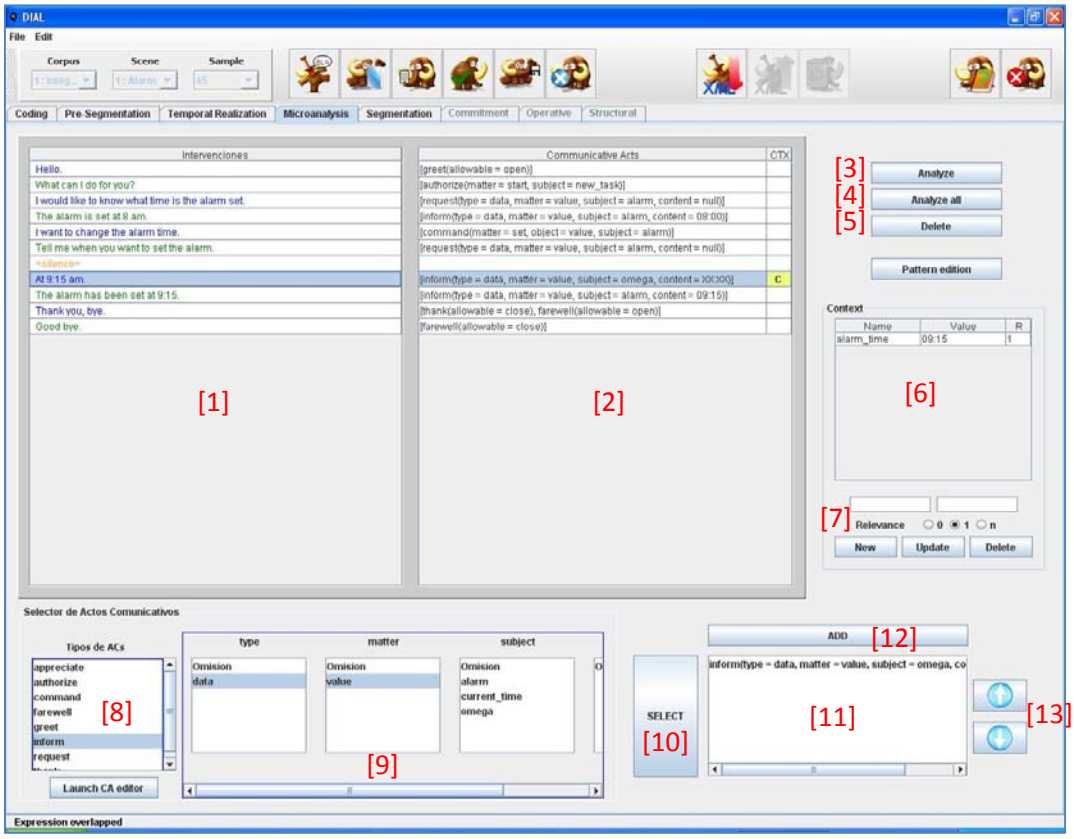

Ilustración 4.13: Pestaña de Microanálisis

Los actos comunicativos, tal y como se ha definido en los requisitos de la fase de microanálisis (ver apartado 3.1.2.3) pueden ser asignados automática o manualmente. Para incluir la funcionalidad automática basta con dos botones ([3] y [4]) que permitirán analizar una expresión o analizar todas, respectivamente.

Para asignar un acto comunicativo manualmente y definir los valores deseados para cada una de sus categorías, se incluirá en la parte inferior de la pestaña una lista de actos comunicativos [8] y a su derecha un panel para las listas de categorías [9]. Es necesario que este panel contenga un *scrollbar* puesto que el número de categorías puede variar dependiendo del acto comunicativo seleccionado. Los actos comunicativos con valores concretos seleccionados para sus categorías se añadirán a otra lista [11] mediante un botón de selección [10]. En esa lista irán apareciendo todos los actos comunicativos

concretos que desean ser asignados posteriormente a una intervención, acción que se realizará mediante otro botón de la interfaz [12].

Por último, para crear variables de contexto y mostrar las pertenecientes a una expresión, es necesario incluir una tabla [6] de tres columnas (nombre, valor y relevancia), dos campos de texto bajo la tabla (para introducir nombre y valor) y tres botones de radio para seleccionar la relevancia (cuyos valores están fijados en 0, 1 o n). Para ayudar a identificar qué expresiones (y actos comunicativos correspondientes) tienen variables de contexto, se añadirá una pequeña columna a la derecha de la lista de actos comunicativos que mostrará una letra (C) en caso de existir.

4.4.5 Interfaz 4: Segmentación

La interfaz de usuario que debe definirse para la fase de segmentación debe recoger en una vista general las expresiones junto con los segmentos creados y las secuencias asignadas a cada expresión. Como se puede apreciar en la Ilustración 4.14, el caso de las expresiones se resolverá introduciendo una tabla [2] o lista que las recoja, en cambio, para el caso de los segmentos existen algunas complicaciones.

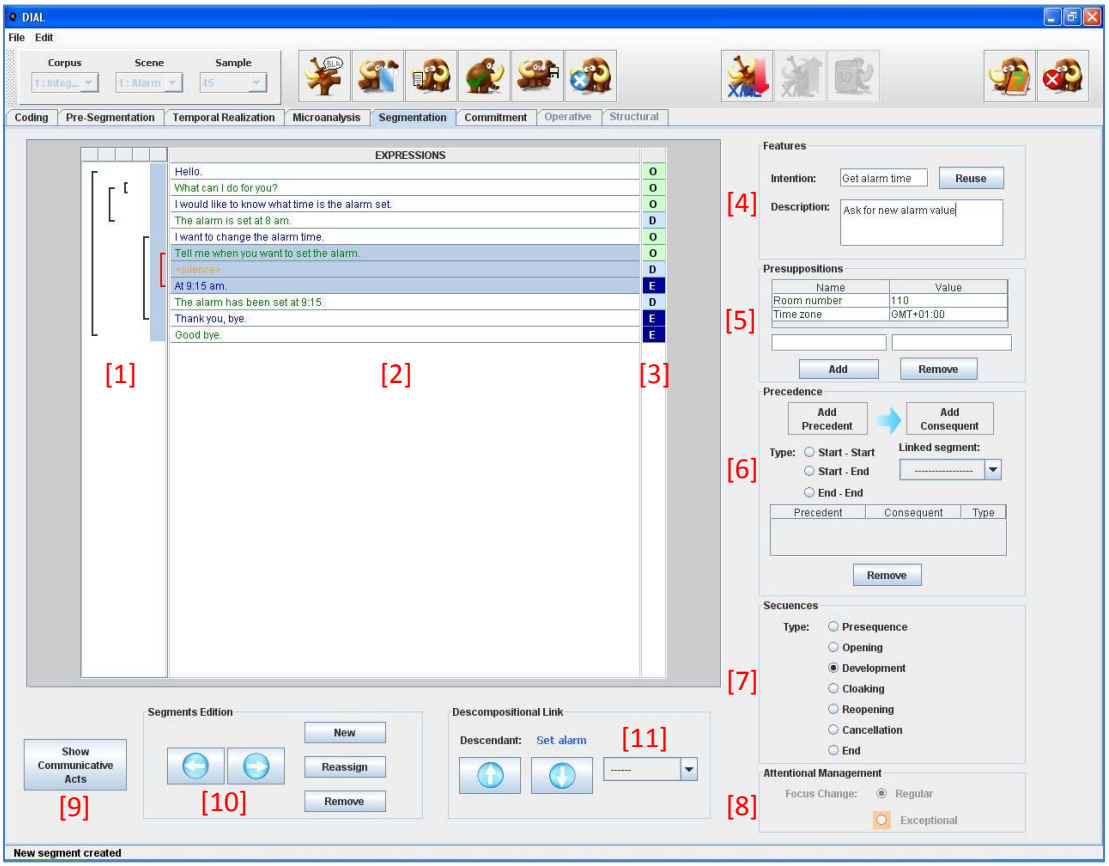

Ilustración 4.14: Pestaña de Segmentación

En primer lugar, esta interfaz debe permitir crear y borrar dinámicamente nuevos segmentos que engloben un conjunto de expresiones. Una vez estudiado el caso, la solución reside en incluir otra tabla [1] situada a la izquierda de las expresiones cuyas

columnas representarán los segmentos. Esta tabla se manejará a nivel de columna (la separación entre filas quedará oculta) y se permitirá tanto la creación como el borrado de columnas. Es importante destacar que como máximo se podrán observar diez segmentos, por lo cual la tabla de segmentos contendrá un *scrollBar* horizontal.

El caso de las secuencias es más sencillo: bastará con añadir una pequeña columna o tabla [3] situada a la derecha de las expresiones en la que se mostrará la inicial y el color de fondo definido para cada tipo de secuencia.

El resto de objetos gráficos que contendrá esta interfaz no entrañan tanta complejidad, puesto que se tratará de botones, tablas, botones de radio, combos de selección o cajas de texto. Dichos controles se repartirán en paneles dependiendo del tipo de función que realicen. En total habrá siete paneles:

- **Panel de edición de segmentos** [10]: Contendrá botones para crear un nuevo segmento, borrar uno determinado, reasignar un nuevo conjunto de expresiones a un segmento y mover un segmento a la derecha o a la izquierda de la tabla.
- **Panel de vínculos descomposicionales** [11]: En este panel se indicará la intención del segmento padre al seleccionado y se permitirá aumentar su nivel descomposicional mediante un botón y disminuirlo mediante un combo de selección (donde los ítems serán las intenciones de los segmentos hermanos) y otro botón.
- **Panel de características generales de un segmento** [4]: Para añadir la intención y características de un segmento se incluirán en este panel dos campos de texto editables.
- **Panel de presuposiciones** [5]: Cuando se seleccione un segmento, deben mostrarse las presuposiciones asignadas al mismo. Para ello bastará con una pequeña tabla de dos columnas (nombre y valor), dos campos de texto para crear una presuposición nueva y dos botones que realizarán las funciones de creación y borrado.
- **Panel de vínculos de precedencia** [6]: En este panel se realizará la gestión de los vínculos de precedencia de un segmento, incluyendo creación, borrado y visualización de los mismos. Para la creación es necesario indicar el tipo de precedencia (como son tres los tipos posibles, se añadirán tres botones de radio), el segmento vinculado (como su número e intención son variables se incluirá un combo de selección) y dos botones para seleccionar si se trata de un precedente al segmento actual o un consecuente. Para mostrar todos los vínculos de un segmento bastará con añadir una tabla de tres columnas (precedente, consecuente y tipo). Por último se añadirá un botón para eliminar un vínculo.

- **Panel de secuencias** [7]: Este panel tiene como función permitir asignar un tipo de secuencia a una expresión. Como existen siete tipos de secuencia definidos, los componentes más indicados son los botones de radio y al poder asignarse solamente un tipo de secuencia para una expresión dichos botones deberán de ser de selección exclusiva.
- **Panel de gestión atencional** [8]: Este pequeño panel permitirá definir si existe o no un cambio de foco excepcional en una expresión, para ello contará con dos botones de radio que definirán un cambio de foco excepcional o regular, respectivamente. Para reconocer en el panel de expresiones en cuales de ellas se produce un cambio de foco excepcional, se utilizará un color vistoso (por ejemplo, naranja) en la expresión de la tabla con el salto excepcional.

Para finalizar, en esta interfaz es posible representar en la misma tabla de expresiones los actos comunicativos correspondientes. Para ello es necesario incluir un botón de tipo *toggleButton* [9] para elegir qué elementos desean mostrarse según las preferencias del usuario.

5 Implementación

En este apartado se procederá a realizar un estudio en profundidad de los principales aspectos de la implementación del sistema. Los detalles y decisiones de implementación que se explicarán tienen como objetivo satisfacer las necesidades funcionales que han sido definidas previamente en el apartado de Análisis y diseño.

En primer lugar se especificará la implementación física del sistema, explicando cómo utiliza los recursos especificados en los requisitos y exponiendo los beneficios que alcanzan las decisiones que han sido tomadas.

En segundo lugar se detallará en profundidad la implementación de los aspectos más relevantes del sistema cliente. Para ello, se hará uso de los diagramas de clases del sistema, definidos por cada uno de los paquetes de implementación (que corresponden con cada componente analizado en el apartado de Arquitectura funcional del sistema). Dentro de cada diagrama de clases, se profundizará en el diseño de información realizado a bajo nivel, describiendo para cada clase del diagrama, sus atributos y métodos más relevantes.

La aplicación se ha desarrollado en el entorno de programación para Java NET BEANS 6, el cual proporciona una mejor gestión del diseño de interfaces, que es la carga principal de este proyecto. El hecho de utilizar un entorno de desarrollo para la implantación de aplicaciones software agiliza la ejecución de pruebas, facilita la depuración y automatiza la compilación, proporciona a su vez utilidades gráficas para la inclusión de librerías externas y control de versiones.

Se ha instalado un servidor del controlador de versiones (Collabnet Subversion) cuyo uso principal es el de realizar copias de seguridad semi-automáticas ya que nunca fue necesario retroceder a versiones anteriores del sistema, aunque es una función de soporte que otorga seguridad al desarrollo.

5.1 Arquitectura física del sistema

Se especificó previamente (en el apartado 4.1) que el sistema implementa una arquitectura cliente-servidor la cual está compuesta con tres máquinas servidoras.

Uno de los servidores contiene la base de datos central que es poblada de conocimiento por las diferentes instancias de la aplicación. Otro servidor contiene la

base de datos en la cual se almacenan los ficheros XML generados tras completar el proceso de análisis de cada una de las muestras de diálogo (conocimiento individual). Por último, existe otro servidor (cuyo uso queda al margen de este proyecto) encargado de almacenar el conocimiento extraído del análisis global de corpus y que será utilizado por una expansión de la herramienta, que se denominará Cognos.Dial Global.

Esta arquitectura está implementada mediante una máquina que ejecuta el programa servidor de la base de datos la cual está localizada en el centro de cálculo de la Universidad Carlos III de Madrid y es propiedad del grupo LABDA. Dispone de una IP versión 4 estática.

Es posible que existan dificultades en cuanto a la conexión del servidor dependiendo de la red en la que se encuentren. Oracle escucha por el puerto 1521 para recibir las conexiones a las distintas bases de datos que tenga. Por tanto, es necesario configurar todos los elementos de bloqueo de conexión (firewalls) del servidor para que permita el acceso a dicho puerto del servidor.

5.2 Descripción detallada de clases

En este apartado de la implementación se definirán los diagramas de clases en UML de cada uno de sus componentes (por cada uno de los paquetes principales del sistema: modelo, vista y controlador). Las ilustraciones de todos los diagramas de clases que aparecerán a lo largo de este apartado se han realizado mediante el programa (Altova UModel).

Es importante señalar que en los diagramas de clases que se mostrarán en los siguientes subapartados únicamente aparecerán los atributos y los métodos más relevantes, de tal manera que el lector llegue a comprender las principales decisiones de implementación que afectan a la funcionalidad del sistema. Por lo tanto, se excluirán de los diagramas de clases aquellos atributos y métodos que no se considere que realicen ninguna aportación relevante para la comprensión de la implementación del sistema.

Por otro lado, a la hora de definir los métodos, se omitirán tanto los parámetros que reciben como el objeto que devuelven. La inclusión de estos detalles supondría un aumento de tamaño considerable en la presentación de los diagramas de clases, dificultando sobremanera tanto la lectura del diagrama como su comprensión. Por este motivo, se ha decidido definir los parámetros y las devoluciones (siempre que se consideren de gran relevancia) dentro de la descripción de cada método y no dentro del diagrama UML.

5.2.1 Paquete Model

El paquete Model contiene la conexión a la base de datos y los mecanismos para acceder y almacenar todos los elementos de los que se compone el sistema. Para ello contiene cuatro componentes básicos: el componente de conexión, indispensable, y los

tres componentes de datos agrupados según la parte de la base de datos utilizan (CAModel, PatternModel e Identification) para así permitir su uso independiente en diferentes herramientas. En este proyecto, sólo se describirá la implementación del paquete View.Identification, puesto que View.CAModel, y View.PatternModel han sido desarrollados en otro proyecto (*Cognos.NL*).

5.2.1.1 Model.Conection

Como se muestra en la Ilustración 5.1, este paquete únicamente contiene una clase llamada DBConection que implementa el patrón de programación Singleton. Este diseño garantiza que sólo existirá en el sistema una instancia de esta clase ya que sólo se necesita una conexión a la base de datos por cada aplicación cliente. Esta clase contiene, como se aprecia en la Ilustración 5.1, un constructor privado *DBConection()* y un atributo del mismo tipo *_conection*. Las clases que requieran contener una instancia de esta clase realizarán una llamada al método *getInstance()*, que devuelve una nueva instancia si no ha sido creada (llamando localmente al constructor) o su atributo en caso de que ya haya sido instanciada con anterioridad.

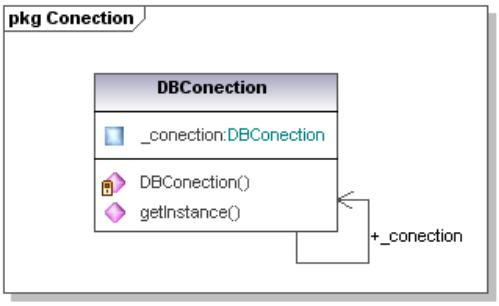

Ilustración 5.1: Conection

Esta clase contiene una conexión a la base de datos que proporciona el *driver* de conexión JDBC que es creada siempre que se invoque al constructor privado, es decir, una única vez por arranque de la aplicación. Esta conexión se crea de la siguiente forma:

```
con = DriverManager.getConnection (URL, username, password);
```

Donde URL es la cadena de host de conexión a la base de datos que en este caso es "jdbc:oracle:thin:@163.117.129.114:1521:DraftCognos". En un futuro se permitirá desconectar y reconectar cambiando estos parámetros mediante alguna interfaz ya que las bases de datos no tienen por qué estar en la misma máquina ni llamarse de la misma forma. Es una línea futura muy deseable de realizar en corto plazo.

Las sentencias SQL que se ejecutarán las elaborarán las clases de los paquetes de información ya que son muy dependientes del objeto y tipo de información que se manipule. Este componente ofrece la posibilidad de ejecutar una sentencia dada y obtener el correspondiente resultado. Se trata, por tanto, de una capa intermedia entre los componentes concretos y la base de datos. El driver de conexión JDBC permite

diferenciar dos operaciones de acceso y manipulación de datos: *executeUpdate()* y *executeQuery()*.

5.2.1.2 Model.Identification

En el diagrama de clases mostrado en la Ilustración 5.2 se observa que existe una clase DB_Identification que sirve de fachada entre las clases concretas que contienen el modelo de datos para la identificación y manipulación de corpus, escenarios y muestras y los paquetes *View* y *Controller*.

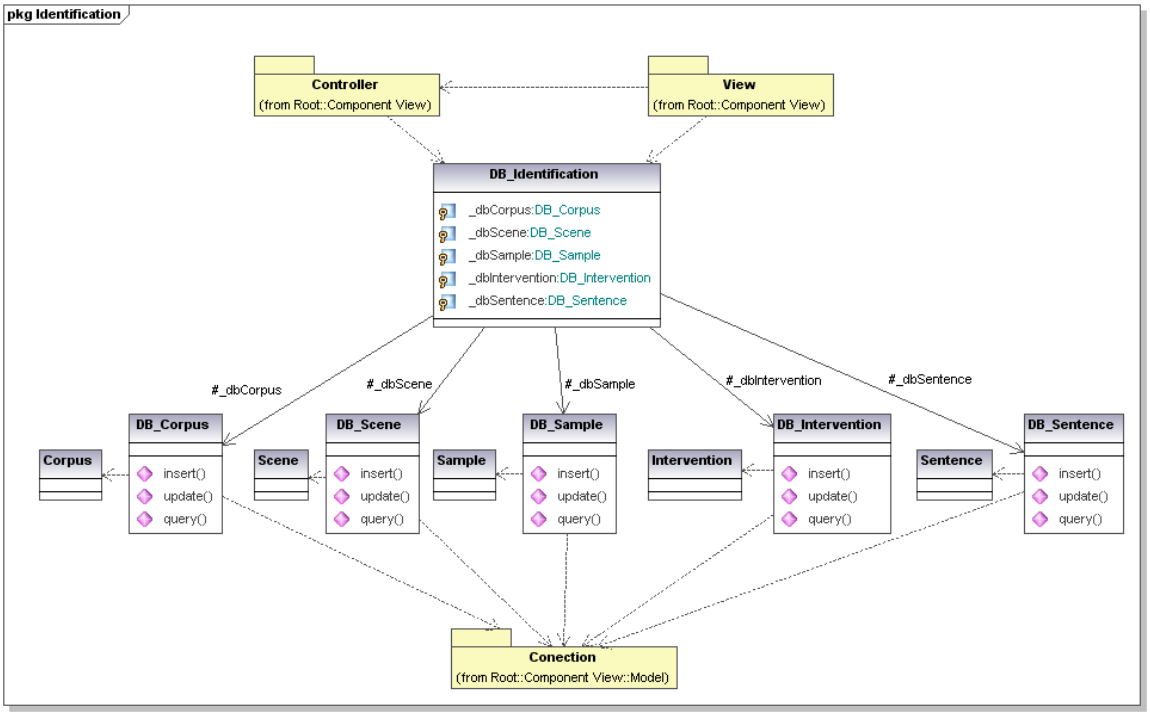

Ilustración 5.2: Identification

Las clases DB_Corpus, DB_Scene, DB_Sample, DB_Intervention y DB_Sentence contienen las operaciones de inserción, actualización y consulta a la base de datos de cada uno de los elementos modelados, respectivamente, por las clases Corpus, Scene, Sample, Intervention y Sentence. Dichas clases utilizan la instancia única de la clase Conection para comunicarse y realizar dichas operaciones con la base de datos.

5.2.2 Paquete View

Este paquete contiene la implementación de todas las interfaces de usuario definidas en el apartado 4.2.2 donde se especifica el análisis y diseño para el paquete Vista.

En la Ilustración 5.4 aparecen todas las clases junto con sus relaciones que se han sido implementadas para los componentes gráficos resultantes del análisis realizado dentro del apartado 4.2.2 (I_CognosDial, I_PreSegmentation, I_TemporalRealization, I_Microanalysis y I_Segmentation).

5.2.2.1 View.I_CognosDial

A continuación se describirá el diagrama de clases que se ha implementado para el componente vista I_CognosDial. Como se puede ver en la Ilustración 5.3, existe una clase principal I_MainFrame que se encuentra relacionada con los demás componentes descritos para el paquete Vista. Cada uno de esos cuatro paquetes que se ven en la ilustración (I_PreSegmentation, I_TemporalRealization, I_Microanalysis e I_Segmentation) corresponde a cada una de las cuatro fases del análisis de corpus de diálogo en las que se centra el proyecto COGNOS.Dial. Para cada una de las fases, existe un diagrama de clases que se comentará en los siguientes apartados.

Dentro de este diagrama de clases, existen otras tres clases relacionadas con I_MainFrame: I_Code, I_Welcome e I_Description. Contienen, respectivamente, la implementación de la interfaz de codificación interna, la interfaz de bienvenida de la aplicación y la interfaz que muestra la descripción tanto de corpus como de escenarios.

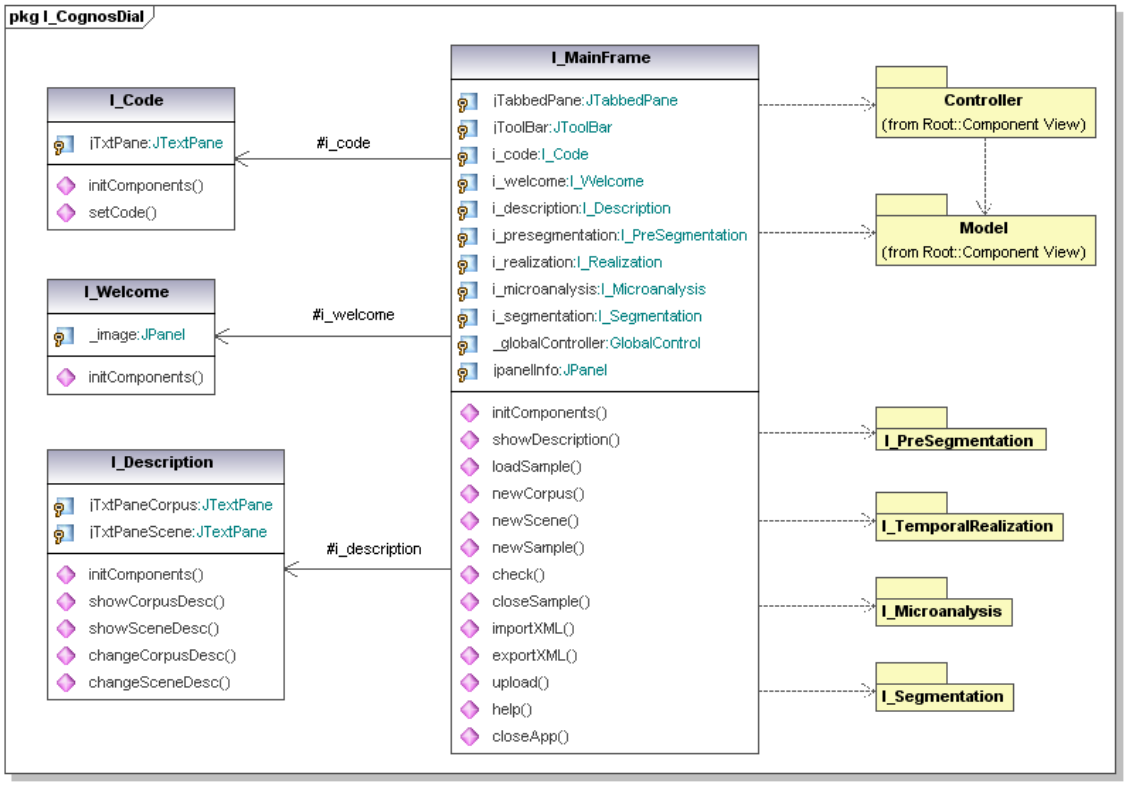

Ilustración 5.3: Diagrama de clases de View.I_CognosDial

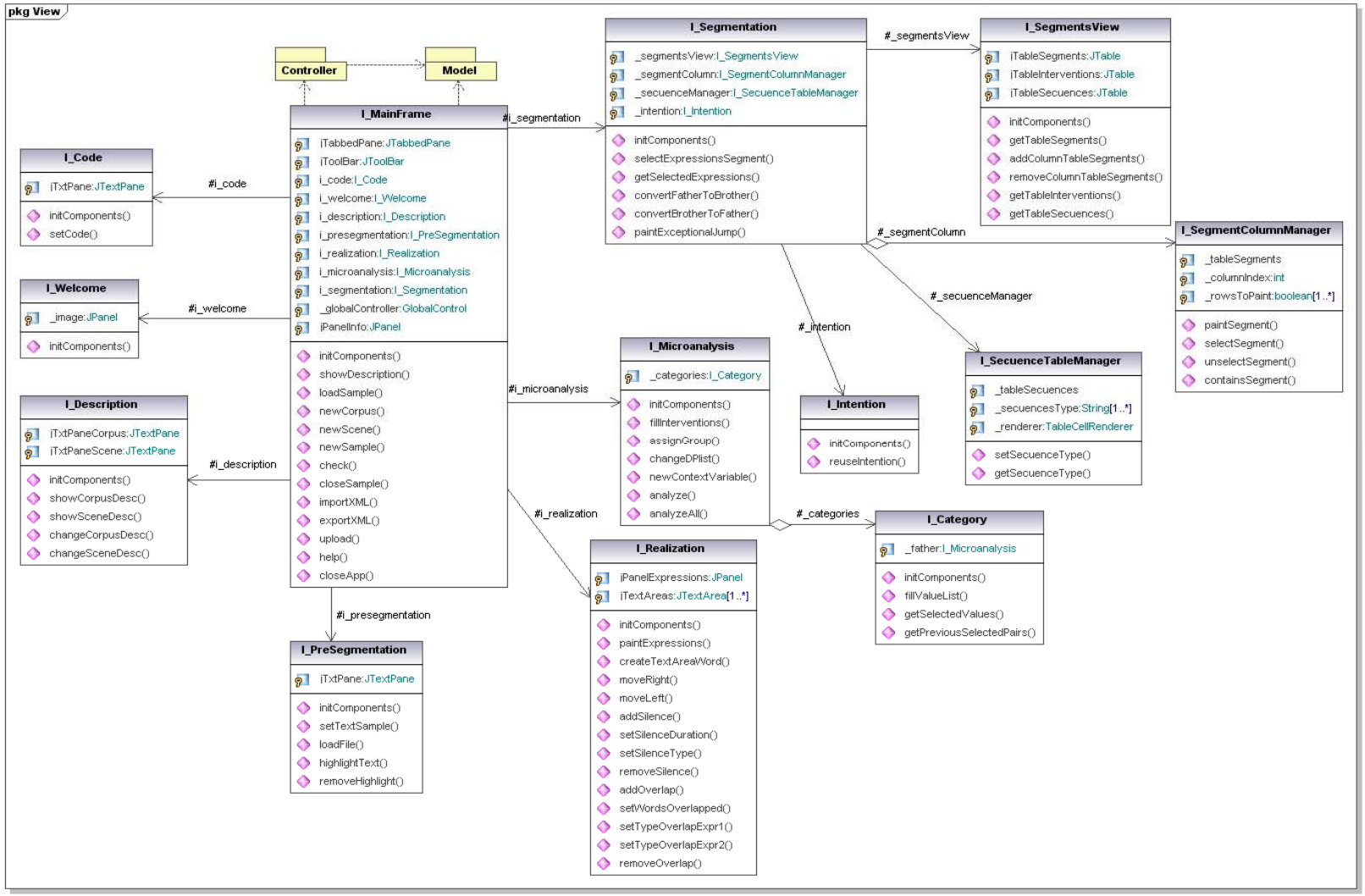

Ilustración 5.4: Diagrama de clases del paquete Vista

A continuación se describirá con más profundidad el contenido de cada una de las clases contenidas en I_CognosDial, haciendo especial hincapié en la clase que implementa la interfaz de usuario principal del sistema: I_MainFrame.

CLASE I_MAINFRAME

En la Ilustración 5.5 se muestran los atributos y métodos principales de la clase principal del paquete Vista: I_MainFrame.

Ilustración 5.5: Clase I_MainFrame

El atributo *jTabbedPane* es un objeto gráfico proporcionado por el paquete gráfico javax.swing que constituye un panel de pestañas. En total, el atributo *jTabbedPane* albergará cinco pestañas, la primera de ellas contiene un objeto de la clase I_Code (atributo *i_code*) que se encarga de representar la interfaz de usuario en la que se puede visualizar el código interno del análisis. La segunda pestaña contiene un objeto de la clase I_PreSegmentation (*i_presegmentation*) que se encuentra implementada dentro del paquete View.I_PreSegmentation y engloba la implementación de la interfaz de usuario de la fase pre-segmentación. Del mismo modo, la tercera, cuarta y quinta pestaña contienen, respectivamente, los atributos *i_realization* (interfaz de usuario para la fase realización temporal), *i_microanalysis* (interfaz de usuario para la fase de microanálisis) e *i_segmentation* (interfaz de usuario para la fase de segmentación).

El atributo *i_welcome* representa simplemente la interfaz de usuario que se muestra al iniciar la aplicación y desaparece una vez que ha comenzado el análisis sobre una muestra.

El atributo *jToolBar* es un objeto gráfico proporcionado por las librerías gráficas de Java y representa la barra de herramientas superior en la cual aparecen los botones comunes a todas las fases del análisis. Dentro de esta barra de herramientas se han incluido tres objetos de tipo JComboBox para permitir la selección de corpus, escenarios y muestras. Además se ha añadido a la barra un total de once objetos JButton desde los cuales se tendrá acceso a las siguientes operaciones: ver la descripción de corpus y escenarios (*showDescription(),* método que hará visible la interfaz *i_description*), cargar una muestra (*loadSample()*), crear nuevos corpus, escenarios y muestras (*newSample()*), chequear una fase del análisis (*check()*), cerrar una muestra (*closeSample()*), importar y exportar una muestra en XML (*importXML()* y *exportXML()*), subir un análisis completado a la base de datos (*upload()*), solicitar ayuda (*help()*) y, por último, cerrar la aplicación (*closeApp()*).

Un atributo imprescindible para que se muestren correctamente los resultados de las operaciones que se han ordenado a través de la interfaz es el denominado *_globalController*, que representa un objeto del paquete Controlador y a través de él será posible acceder a todas las operaciones que tienen que ver con la lógica de la aplicación.

Por último, destacar que el método *initComponents()* se encarga de inicializar todos los componentes gráficos que se han diseñado con el entorno de desarrollo (NetBeans 6.7). Este método es generado automáticamente por la herramienta siempre que se diseña una interfaz gráfica, por lo tanto aparecerá en todos las clases de la aplicación que contengan una interfaz gráfica.

CLASE I_CODE

La clase I_Code que se muestra en la Ilustración 5.6 contiene la implementación de una interfaz gráfica sencilla. En dicha interfaz únicamente existe un panel de texto (atributo *jTxtPane*) dentro del cual se muestra el código interno que se haya generado dependiendo de la fase en la que se encuentre el análisis.

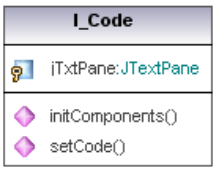

Ilustración 5.6: Clase I_Code

Mediante el método *setCode()* será posible actualizar el contenido del código mostrado en el panel de texto.

Conviene recordar en este punto que en toda la aplicación solamente existe una instancia de la clase I_Code que se corresponde con el atributo existente dentro de la clase I_MainFrame (*i_code*) y que se hace visible a través de la primera pestaña de la herramienta.

CLASE I_WELCOME

La clase I_Welcome mostrada en la Ilustración 5.7 representa sencillamente la interfaz gráfica de bienvenida a la aplicación. El atributo *_image* es un panel que contiene exclusivamente una imagen a modo de presentación que se mostrará en el lugar que ocupa el panel de pestañas antes de que el usuario comience el análisis de una muestra

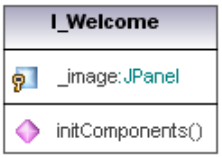

Ilustración 5.7: Clase I_Welcome

CLASE I_DESCRIPTION

La clase I_Description que se muestra en la Ilustración 5.8 se encarga de mostrar una interfaz gráfica. Esta podrá tener dos apariencias diferentes dependiendo de lo que desee consultar el usuario. Si se encuentran seleccionados un corpus y un escenario en los JComboBox de la barra de herramientas principal, la interfaz I_Description muestra las descripciones de ambos haciendo uso de dos paneles de texto (*jTxtPaneCorpus* y *jTxtPaneScene*). Por otra parte, si únicamente se encuentra seleccionado un corpus, la interfaz I_Description contendrá únicamente un panel de texto para mostrar la descripción del corpus (*jTxtPaneCorpus*).

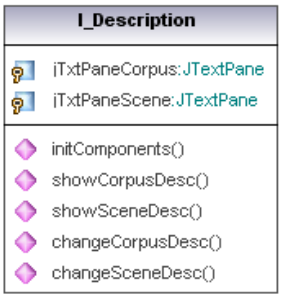

Ilustración 5.8: Clase I_Description

Aunque no aparezcan en la Ilustración 5.8, la clase I_Description cuenta con dos constructores diferentes: el primero contiene dos claves (el identificador de un corpus y de un escenario); el otro constructor contiene solamente una clave (el identificador de un corpus). De este modo, se inicializarán uno o dos paneles de texto y para obtener la descripción que debe de incluirse en ellos se hará uso de los métodos *showCorpusDesc()* y *showSceneDesc()* que se comunican con el paquete Modelo para realizar las consultas necesarias a la BD.

5.2.2.2 View.I_PreSegmentation

El paquete View.I_PreSegmentation es el responsable de albergar la interfaz de usuario necesaria para realizar la fase de análisis de pre-segmentación. Como se puede

apreciar en la Ilustración 5.9: Diagrama de clases de View.I_PreSegmentation, debido a la sencillez de la interfaz, ha sido necesario implementar únicamente una clase denominada I_PreSegmentation. Para que las funciones que son accesibles desde esta clase de interfaz gráfica se ejecuten y se muestre el resultado deseado, es necesaria la comunicación con el paquete Controlador.

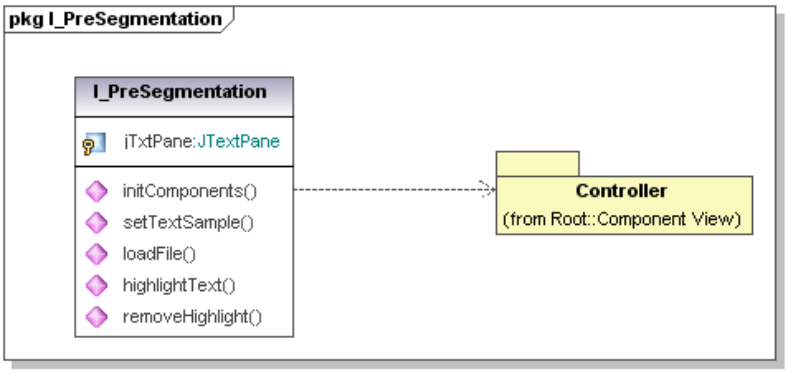

Ilustración 5.9: Diagrama de clases de View.I_PreSegmentation

CLASE I_PRESEGMENTATION

En la Ilustración 5.10 se muestra la estructura de la clase I_PreSegmentation, la cual posee un atributo denominado *jTxtPane* (objeto gráfico de tipo JTextPane proporcionado por el paquete javax.swing) necesario para mostrar el contenido de una muestra de diálogo. Dicho panel de texto es editable por el usuario, siendo posible modificar su contenido parcial o totalmente. Aunque el texto de la muestra se visualizará con formato plano, internamente es almacenado en formato HTML y el tratamiento del mismo se realizará en el paquete Controlador.

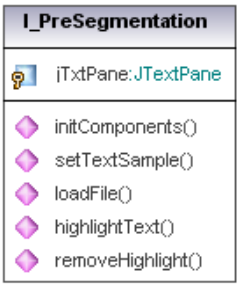

Ilustración 5.10: Clase I_PreSegmentation

El método *setTextSample()* se encarga de insertar el texto de la muestra de diálogo dentro del panel *jTxtPane*. Este texto puede provenir de una muestra almacenada en la base de datos y que ha sido seleccionada en la barra de herramientas o de un fichero externo a la aplicación que se encuentra en la máquina donde está instalada la aplicación cliente, para lo cual se utiliza el método *loadFile()*. También es posible que se cree una nueva muestra vacía, en este caso el método *setTextSample()* no insertará nada en el panel de texto y el usuario podrá escribir el texto directamente en el panel.

Esta clase instancia además un objeto de tipo JComboBox en el que se puede elegir un rol para asignarlo a un fragmento de texto. Cuando se selecciona un fragmento de texto en el panel, el color amarillo de dicha selección debe permanecer mientras se cambia el foco al jComboBox de roles. Por este motivo, ha sido necesario implementar un método *highlightText()* que mantiene dicha selección hasta que se ha elegido uno de los dos roles posibles. No obstante, el posterior cambio de color de fuente del fragmento de texto es responsabilidad del paquete Controlador y se explicará dicho procedimiento en el apartado 4.2.3.1 (Controller.GlobalControlAnalysis).

El método *removeHightlight()* realiza la operación contraria a *highlightText()*, es decir, elimina la selección en amarillo del texto. Este método es invocado cuando, seguidamente a la selección de un fragmento de texto no se elige un rol para el mismo y se procede a realizar cualquier otra operación en la interfaz I_PreSegmentation.

5.2.2.3 View.I_TemporalRealization

El diagrama de clases mostrado en la Ilustración 5.11 representa la única clase contenida dentro del componente View.I_TemporalRealization, la que implementa la interfaz de usuario para la fase de realización temporal.

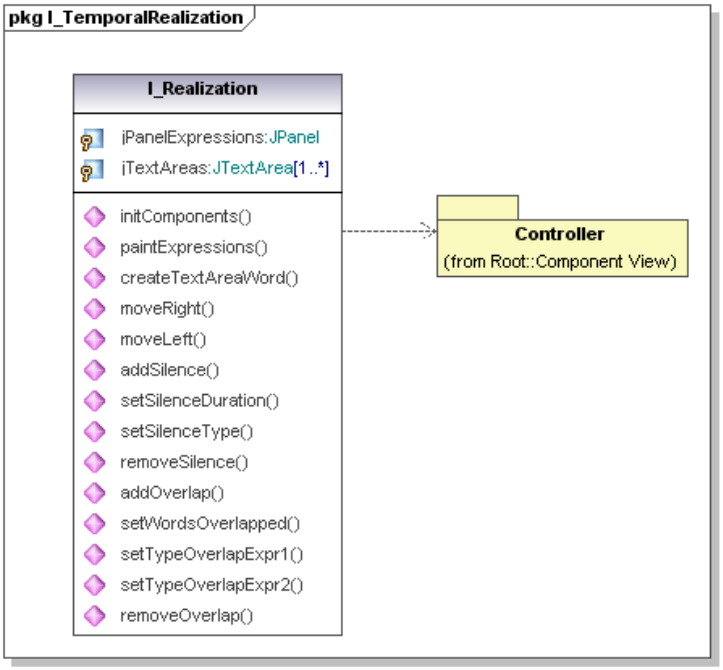

Ilustración 5.11: Diagrama de clases de View.I_TemporalRealization

Al igual que ocurre con la clase I_PreSegmentation, esta clase se comunica con el paquete Controlador el cual procesa la información que le llega de la vista I_TemporalRealization y devuelve el resultado que ha de mostrarse para cada operación.

CLASE I_REALIZATION

La clase I_Realization contiene los objetos gráficos de interfaz necesarios para incorporar la información prosódica que se ha decidido contemplar en el apartado de análisis (4.2.2.3, View.I_TemporalRealization), es decir, silencios y solapamientos que suceden entres dos expresiones consecutivas.

En la Ilustración 5.12 aparece en primer lugar el atributo *jPanelExpressions*, objeto de tipo JPanel que tiene gran relevancia debido a la gran cantidad de operaciones gráficas que se realizan sobre él. La mayor dificultad que entraña la interfaz de usuario en esta fase, es conseguir mover expresiones dentro del panel de forma que se creen espacios en blanco (silencios) o solapamientos entre ellas. La decisión que se ha tomado a la hora de realizar la implementación para solucionar este inconveniente ha sido la de crear una matriz de objetos de tipo JTextArea (atributo *jTextAreas*), de modo que cada fila de la matriz contiene todas las palabras de una expresión dentro de objetos JTextArea. De este modo, será más sencillo conseguir que se desplacen las expresiones dentro del panel y que los solapamientos se realicen en un número determinado de palabras completas.

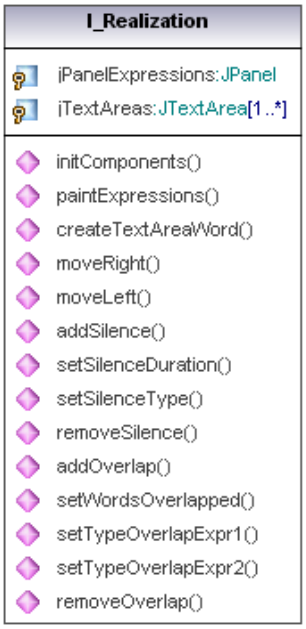

Ilustración 5.12: Clase I_Realization

Para crear un área de texto a medida para una palabra se ha implementado el método *createTextAreaWord()*. El área de texto creada tendrá un color de fondo que corresponde con el rol de la expresión a la que pertenece la palabra. El método *paintExpressions()* se encarga de recorrer la matriz *jTextAreas* y por cada una de sus filas (cada fila contiene las áreas de texto de una expresión) pintará dentro de *jPanelExpressions* las áreas de texto de todas las palabras una a continuación de otra (de modo que todas las palabras de una expresión se muestren consecutivamente). Cada

expresión se pintará en una línea diferente del panel y cada una de ellas comenzará en el punto donde acaba la anterior (importante para poder representar los solapamientos).

Una vez que las expresiones están incluidas en *jPanelExpressions*, es posible desplazar cada una de ellas hacia la derecha (para crear un silencio) o hacia la izquierda (para crear un solapamiento de una palabra). Para ello, se han implementado los métodos *moveRigth()* y *moveLeft().*

Para representar un silencio, se dibujará un pequeño área de texto dentro del panel e inmediatamente después de la expresión que da paso al mismo. Éste contendrá la inicial que corresponde al tipo de silencio elegido y un color que lo distingue. De realizar estas operaciones se encarga el método *addSilence()*. A la hora de modificar el tipo de un silencio, se invocará al método *setSilenceType()* el cual cambiará el área de texto correspondiente al tipo de silencio anterior e incluirá en la misma posición del panel el área de texto correspondiente al nuevo tipo. Por otro lado, un silencio al ser creado tiene una duración por defecto de 0,5 segundos pero es posible aumentar la duración del mismo mediante el método *setSilenceDuration()*. Para visualizar ese aumento de duración, detrás del área de texto de silencio se añadirán tantos guiones como unidades de silencio se introduzcan (tomando como granularidad temporal la cantidad 0,5 segundos).

Los solapamientos físicos entre dos expresiones son posibles siempre que ambas sean de diferente rol, si esto no ocurre, podrá añadirse un solapamiento pero no existirá ninguna palabra solapada entre las expresiones involucradas. El método *addOverlap()* añade un solapamiento (ya sea físico o no) entre dos expresiones. En caso de introducirse un solapamiento físico, realiza la misma operación que el método *moveLeft(),* es decir mueve todas las áreas de texto de la expresión solapada (y las siguiente expresiones) una palabra hacia la izquierda. Para aumentar el número de palabras solapadas, se hace uso del método *setWordsOverlapped().* Para modificar el tipo de cambio de tono que realizan tanto el hablante interrumpido como el que interrumpe se han implementado los métodos *setTypeOverlapExpr1()* y *setTypeOverlapExpr2().*

Para eliminar tanto silencios como solapamientos, se harán llamadas a los métodos *removeSilence()* y *removeOverlap()*.Ambos, eliminan una unidad de silencio o de solapamiento respectivamente y borran del panel de expresiones la marca existente.

5.2.2.4 View.I_Microanalysis

El componente View.Microanalysis implementa la interfaz gráfica de la fase de microanálisis. En el diagrama de clases que se puede apreciar en la Ilustración 5.13, existe una clase interfaz principal I_Microanalysis, que contiene objetos gráficos de la clase I_Category. Ambas serán descritas en detalle a continuación.

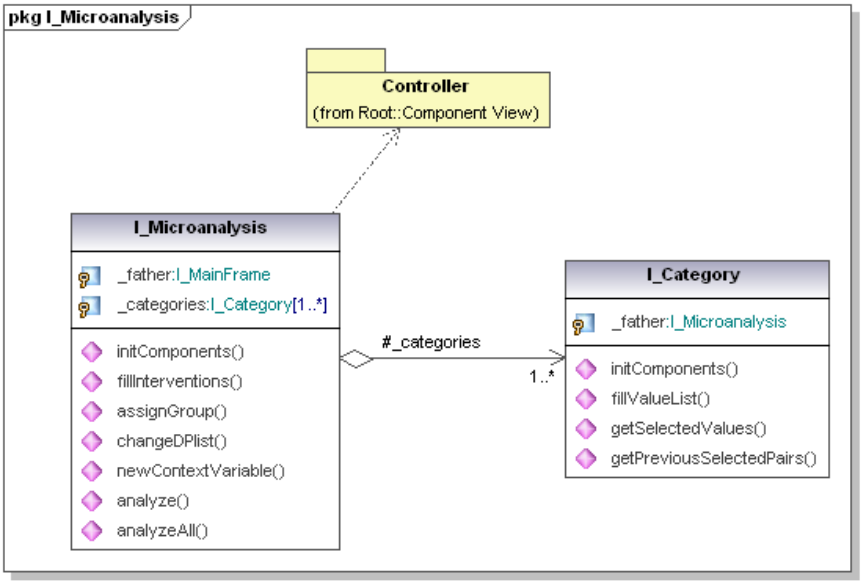

Ilustración 5.13: Diagrama de clases de View.I_Microanalysis

CLASE I_MICROANALYSIS

La clase I_Microanalysis mostrada en la Ilustración 5.14 contiene un atributo *_categories* que representa un conjunto de objetos de la clase I_Category que corresponden a los paneles de categoría que se muestran por cada acto comunicativo seleccionado.

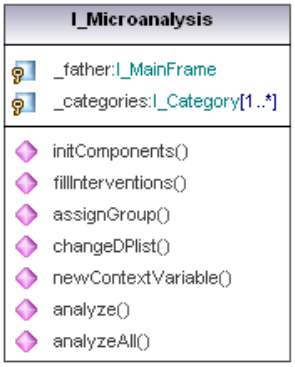

Ilustración 5.14: Clase I_Microanalysis

El método *fillInterventions()* se encarga de rellenar la tabla de expresiones que provienen del paquete Controlador. Por otro lado, *assignGroup()* asigna a una expresión el de conjunto de actos comunicativos que se encuentra presente en la lista de actos comunicativos concretos seleccionados. Para volver a cargar la lista de actos comunicativos seleccionados de modo que se visualicen aquellos pertenecientes a la expresión seleccionada se ha implementado el método *changeDPlist()*.

Desde esta interfaz es posible añadir variables de contexto. Para este fin, se incluye el método *newContextVariable()*, que añade una nueva variable a la tabla.

Por último los métodos *analyze()* y *analyzeAll()* contienen las llamadas correspondientes al paquete Controlador para realizar el análisis de una o de todas las expresiones.

CLASE I_CATEGORY

La clase I_Category mostrada en la Ilustración 5.15 es la encargada de mostrar la información de cada categoría de un acto comunicativo concreto. La funcionalidad de este componente es dependiente de los objetos que le contengan, ya que para mostrar los valores de una cierta categoría es necesario conocer los valores anteriores seleccionados anteriormente. De esto se encarga el método *getPreviousSelectedPairs()* el cual obtiene los datos sobre los paneles anteriores que posee el objeto que está usando cada instancia concreta. Los valores que muestra esta clase en una JList son los válidos para los pares seleccionados previos.

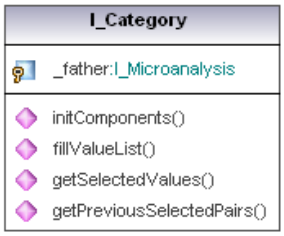

Ilustración 5.15: Clase I_Category

En el momento en el que la clase I_Microanalysis añada el acto comunicativo, se obtendrán los valores seleccionados para cada panel de categoría y se formará el String que representa el acto comunicativo concreto.

5.2.2.5 View.I_Segmentation

En la Ilustración 5.16 se muestra el diagrama de clases implementado para el componente View.I_Segmentation, que contiene todos los elementos de interfaz de usuario necesarios para realizar la fase de segmentación.

En esta fase se realizan las operaciones de segmentación, secuenciación y gestión atencional sobre la muestra, por lo cual es necesario que la interfaz ofrezca una visión global de los segmentos, secuencias y saltos excepcionales creados. Para conseguir apreciar a simple vista los segmentos junto con todas las expresiones de la muestra y sus respectivas secuencias se ha implementado la clase I_SegmentsView. Las clases I_SegmentColumnManager y I_SecuenceTableManager son utilizadas para la representación gráfica de segmentos y secuencias respectivamente. Tanto estas dos clases como I_SegmentsView son manejadas desde la vista general de la fase que se encuentra implementada en I_Segmentation.

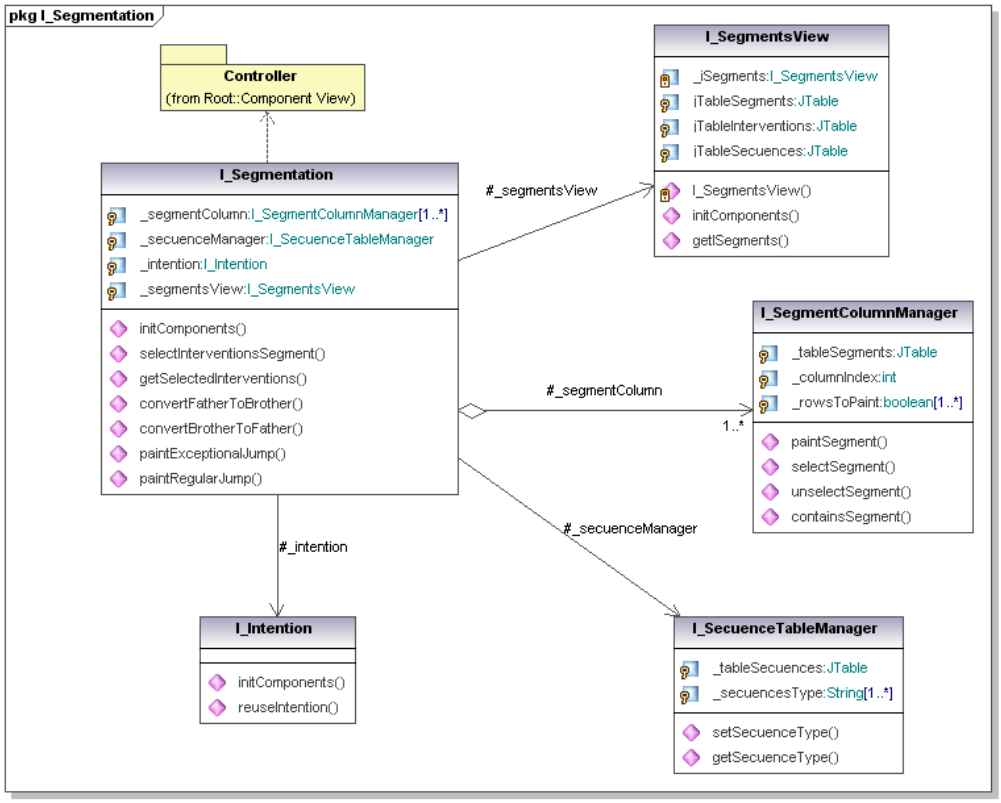

Ilustración 5.16: Diagrama de clases de View.I_Segmentation

A continuación se explicaran los atributos y métodos principales de cada una de las clases de View.I_Segmentation.

CLASE I_SEGMENTATION

Esta clase contiene un atributo denominado *_segmentsView* el cual contiene la vista general de las expresiones junto con los segmentos y las secuencias. Mediante el vector de elementos *_segmentColumn* será posible manejar la apariencia de los segmentos (cada uno de ellos corresponde con una columna en *_segmentsView*).

La visualización de las secuencias correspondientes a cada expresión se modificará gracias al atributo *_secuenceManager*. El último atributo que se muestra en la Ilustración 5.17, I_Intention representa una pequeña interfaz que permite reutilizar una intención (identificador de un segmento) perteneciente al mismo corpus y almacenada en la base de datos. Aunque esta interfaz gráfica ya está disponible, su funcionalidad completa queda como línea futura.

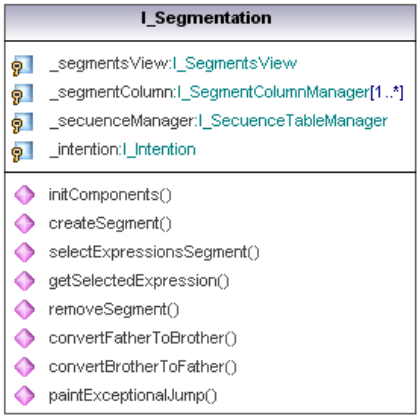

Ilustración 5.17: Clase I_Segmentation

Aunque no aparecen en el diagrama de clases, puesto que su implementación es sencilla, (gracias a la utilización del diseñador de interfaces que proporciona el entorno NetBeans) dentro de la clase I_Segmentation existen numerosos objetos gráficos, tales como botones simples, tablas, campos de texto, botones de opción y selectores que permiten manejar la edición de un segmento, vínculos descomposicionales, presuposiciones y vínculos de precedencia, así como definir los tipos de secuencia y los tipos de salto para la gestión atencional.

El método *createSegment()* se encarga de ordenar la creación de una nueva columna en *_segmentsView* indicándole las expresiones que debe contener (lo cual se obtiene mediante *getSelectedExpressions()*). Cuando se selecciona un segmento, se seleccionan automáticamente las expresiones que contiene, para lo que se ha implementado el método *selectExpressionsSegment()*. El método *removeSegment()* elimina un segmento (y por lo tanto una columna) creado previamente (a excepción del segmento base). Hay que señalar que al eliminar un segmento, se borra toda la información perteneciente a éste, pero no las intervenciones asociadas al mismo.

Los métodos *convertFatherToBrother()* y *convertBrotherToFather()* han sido implementados para permitir modificar el vínculo descomposicional de modo que se permita modificar el padre de un segmento (estos métodos se encargan de modificar la estructura del segmento). Por último, el método *paintExceptionalJump()* colorea de naranja el fondo de la expresión en la cual se produce un salto expcepcional.

CLASE I_SEGMENTSVIEW

La clase I_SegmentsView que se aprecia en la Ilustración 5.18 contiene los objetos gráficos implementados para permitir visualizar el conjunto de expresiones de la muestra junto con la totalidad de los segmentos identificados y el tipo de secuencia correspondiente a cada expresión.

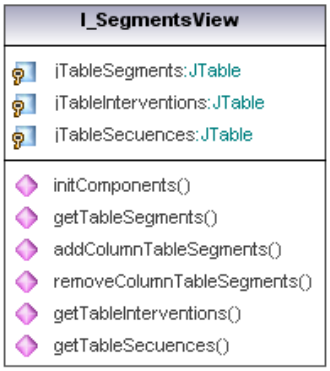

Ilustración 5.18: Clase I_SegmentsView

Para conseguir visualizar segmentos, expresiones y secuencias, se han creado tres tablas independientes: *jTableSegments*, *jTableInterventions* y *jTableSecuences*. Aunque se trate de tres objetos JTable diferentes, visualmente aparecen como si se tratasen conjuntamente de una sola tabla.

Las tablas *jTableInterventions* y *jTableSecuences* mantendrán siempre la misma estructura: una columna y tantas filas como expresiones diferentes se hayan identificado en la muestra. En cambio, *jTableSegments* tendrá inicialmente una columna (correspondiente al segmento base) pero a medida que se vayan creando segmentos, se irán añadiendo nuevas columnas hacia la izquierda (mediante el método *addColumnTableSegments()*). El número de filas de todas las columnas que se añadan en *jTableSegments* será el mismo que el de las otras dos tablas. Cuando se desee eliminar un segmento será necesario invocar el método *removeColumnTableSegments()*. Para que dichas tablas sean accesibles desde las demás clases, se han implementado los métodos *getTableSegments()*, *getTableInterventions()* y *getTableSecuences()*.

CLASE I_SEGMENTCOLUMNMANAGER

La clase I_SegmentColumnManager maneja la visualización de un segmento (una columna de *_tableSegments*). El atributo *_columnIndex* indica el orden en el que ha sido creado el segmento. El último atributo mostrado en la Ilustración 5.19 (*_rowsToPaint*) representa las filas de la columna que deben ser pintadas de modo que se aprecie visualmente qué expresiones están contenidas en el segmento. Para facilitar esta labor, se ha decidido que *_rowsToPaint* consista en un array de booleanos, de tal forma que cada elemento del array corresponde con una fila de la columna, siendo true si esa fila debe ser pintada (la expresión está contenida en el segmento) y false en caso contrario.

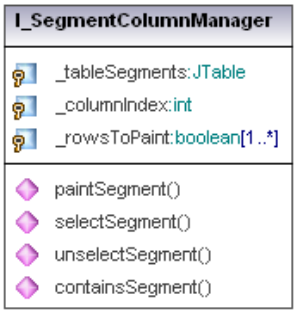

Ilustración 5.19: Clase I_SegmentColumnManager

El método *paintSegment()* recorre el array *_rowsToPaint()* e incluye en las filas que contengan el valor true un símbolo de uno de los tipos que recoge la Ilustración 5.20. La finalidad que se persigue con la inclusión de estos símbolos es que se forme un corchete que englobe las expresiones contenidas en el segmento.

┌	Inicio de un segmento
│	Interior de un segmento
└	Fin de un segmento
:	Oscurecimiento de un segmento
[	Segmento con una sola expresión

Ilustración 5.20: Símbolos para la visualización de segmentos

Los métodos *selectSegment()* y *unSelectSegment()* mantienen y eliminan, respectivamente, la selección de un segmento. El color de fuente de los símbolos definidos en la Ilustración 5.20 será rojo cuando un segmento se encuentre seleccionado. Al añadir la precedencia de un segmento, es necesario que se mantengan dos segmentos seleccionados. Por lo tanto, en este caso y para diferenciarlos, los símbolos del segundo segmento serán de color amarillo. Por último, el método *containsSegment()* comprueba si un segmento está contenido totalmente dentro de otro y es utilizado para identificar el padre de un segmento.

CLASE I_SECUENCETABLEMANAGER

La clase I_SecuenceTableManager maneja el contenido de la tabla *_tableSecuences* que contiene tantas filas como expresiones haya en la muestra. Como muestra la Ilustración 5.21 , contiene un atributo denominado *_secuencesType* consistente en un array de objetos String. Dicho array contiene tantas posiciones como filas contiene *_tableSecuences* y contiene en cada posición la inicial del tipo de secuencia de cada expresión. Además será utilizado por el atributo *_renderer* para pintar cada celda de *_tableSecuences* de un determinado color dependiendo del tipo de secuencia que contenga (Ver el análisis de View.I_Segmentation en el apartado 4.2.2.5).

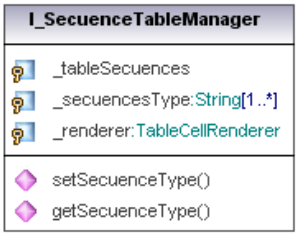

Ilustración 5.21: Clase I_SecuenceTableManager

Con el método *setSecuenceType()* se modificará el tipo de secuencia de una expresión y mediante *getSecuenceType()* se consultará el tipo de secuencia de una expresión determinada.

CLASE I_INTENTION

La clase I_Intention mostrada en la Ilustración 5.22 representa una pequeña interfaz que se hace visible en el momento en el que se desea reutilizar una intención ya existente en el mismo corpus al que pertenece la muestra que está siendo editada y que se encuentra almacenada en la base de datos. La funcionalidad de esta interfaz aún no se ha desarrollado completamente, y quedará como línea futura.

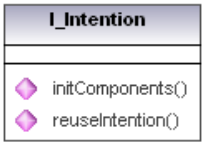

Ilustración 5.22: Clase I_Intention

En estos momentos sólo se encuentra disponible la interfaz de usuario y se pretende que el método *reuseIntention()* (en desarrollo) complete la funcionalidad de la interfaz realizando una petición al controlador que consulte en la base de datos los datos de las intenciones almacenadas.

5.2.3 Paquete Controller

El paquete Controller se encarga de recibir las solicitudes enviadas través de la interfaz de usuario y procesar la información recibida, realizando las operaciones y aplicando los algoritmos pertinentes que tienen como fin almacenar estructuras de información temporales (empleando información solicitada al paquete Model) y devolver una respuesta al paquete View para que éste la muestre al usuario a través de su interfaz gráfica.

Dentro del paquete Controller existen dos componentes: GlobalControlAnalysis y Utilities. El primero de ellos desempeña la función de controlador principal y contiene las principales estructuras de memoria y algoritmos de la aplicación, en cambio el componente Utilities ejerce una función auxiliar ya que proporciona a GlobalControlAnalysis servicios complementarios que permiten la implementación de algunas de sus operaciones.

5.2.3.1 Controller.GlobalControlAnalysis

El diagrama de clases del componente GolbalControlAnalysis que se puede apreciar en la Ilustración 5.23 engloba toda la lógica de negocio de la aplicación. A modo de visión general, es importante destacar la existencia de un controlador global (accesible desde el paquete Vista) que tiene visibilidad y control sobre el resto de clases que están incluidas en el diagrama.

Las clases PresegmentationControl, MycroAnalysisControl y SegmentationControl tratan la información recibida a través de las interfaces de usuario correspondientes a las fases de pre-segmentación, microanálisis y segmentación, respectivamente. Para la fase de realización temporal no se ha creado una clase de control, ya que la información editada a través de la interfaz será procesada mediante la clase Expression y los vínculos existentes con Silence y Overlap.

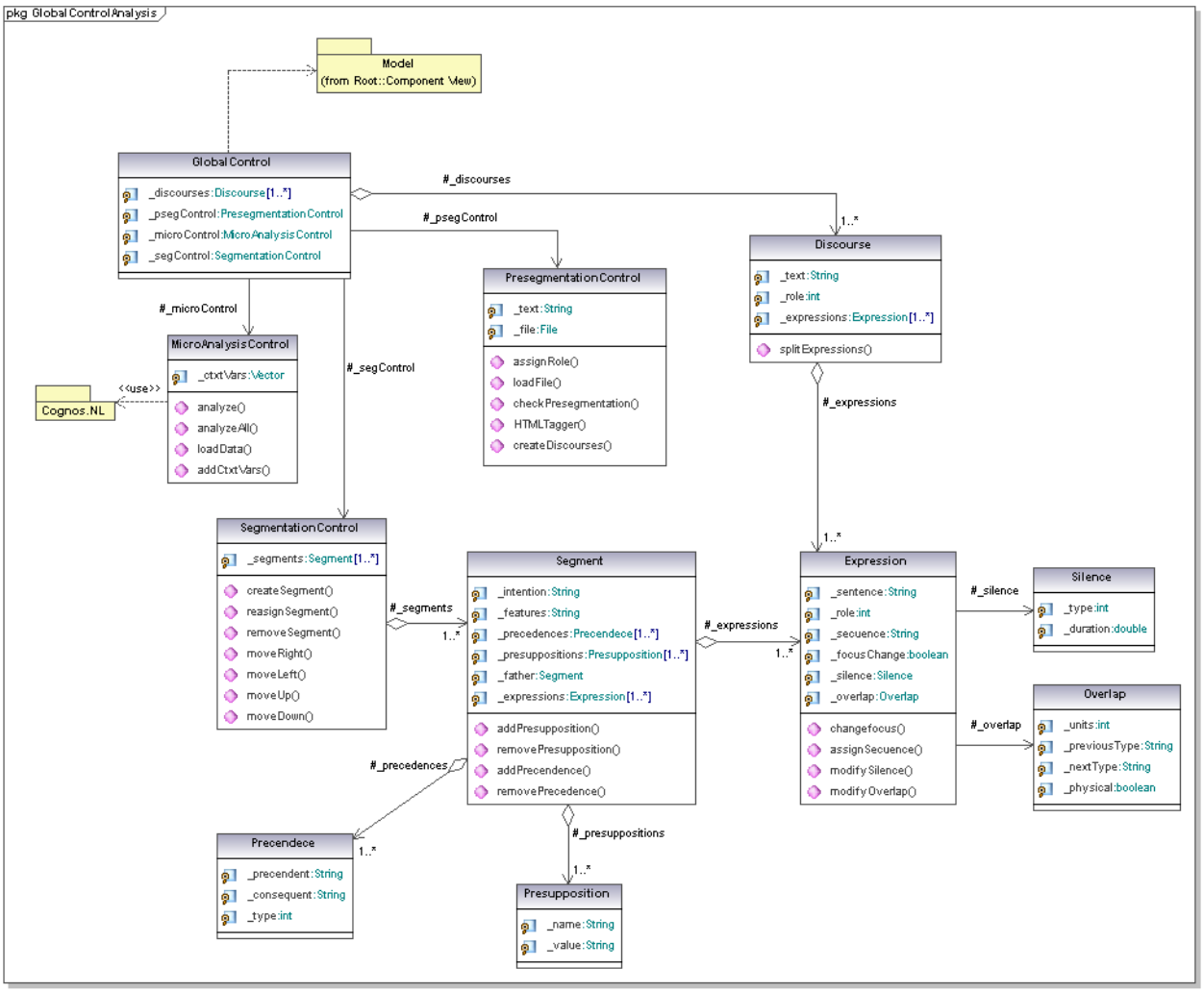

Ilustración 5.23: Diagrama de clases de Controller.GlobalControlAnalysis

Seguidamente se describirá cada una de las clases del diagrama junto con sus principales atributos y funciones.

CLASE GLOBALCONTROL

La clase GlobalControl que se muestra en la Ilustración 5.24 contiene un atributo *_interventions* que representa un vector con todos las intervenciones existentes en la muestra. Este atributo es inicializado una vez que se ha realizado la fase de pre-segmentación. Esta clase representa un acceso global a todo el controlador, gracias a los atributos *_psegControl*, *_microControl* y *_segControl* (controladores de presegmentación, microanálisis y segmentación). Para acceder a los métodos que ofrecen estos controladores así como a la estructura de discursos se han implementado los métodos de acceso *getInterventions()*, *getPSegControl()*, *getMicroControl()* y *getSegControl()*

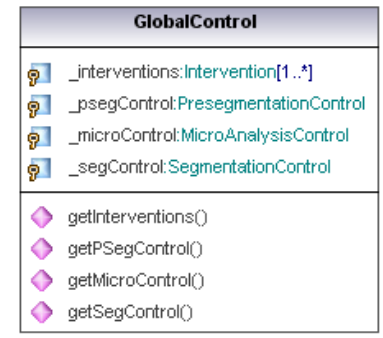

Ilustración 5.24: Clase GlobalControl

CLASE PRESEGMENTATIONCONTROL

La clase PresegmentationControl que se recoge esquemáticamente en la Ilustración 5.25 recoge toda la lógica necesaria para realizar la fase de pre-segmentación de una muestra. Como atributos principales contiene *_text* que corresponde con la totalidad del texto del diálogo y *_file* que representa el fichero externo desde el cual se desea extraer el texto de la muestra (para lo cual se empleará el método *loadFile()*).

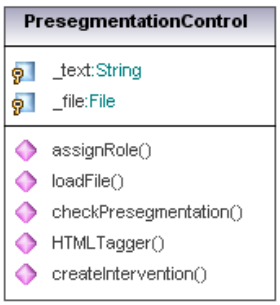

Ilustración 5.25: Clase PresegmentationControl

El método *assignRole()* recibe un fragmento de texto del diálogo y le asigna un rol, lo que conlleva la realización de dos operaciones importantes: por un lado, mediante el método *HTMLTagger()* se etiqueta ese fragmento de texto en HTML con el color de fuente correspondiente al rol asignado (devolviendo el texto completo al paquete Vista); y por otro lado, mediante el método *createIntervention()* se crea un nuevo objeto Intervention con ese fragmento de texto y ese rol.

Por último, en esta clase también se encuentra implementado un método para el chequeo de la fase de pre-segmentación (*checkPresegmentation()*). Este método devuelve un resultado positivo en caso de que todo el texto se encuentre etiquetado con alguno de los dos roles del diálogo.

CLASE DISCOURSE

La clase Intervention que se muestra en la Ilustración 5.26 representa un conjunto de expresiones del mismo rol que suceden consecutivamente en el tiempo. El conjunto de expresiones (array *_expressions*) es determinado por el método *splitExpressions()* que se encarga de identificar cuando hay un cambio de expresión (inicialmente, los cambios se identifican por la aparición del signo punto y aparte o el carácter de retorno de carro). Esta clase contiene además como atributos el texto de la intervención (*_text*) y su rol (*_role*).

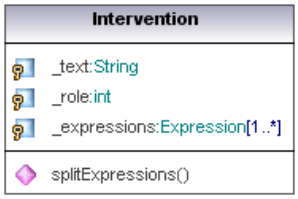

Ilustración 5.26: Clase Intervention

CLASE EXPRESSION

Esta clase representa una expresión de la muestra y será utilizada en cada una de las siguientes fases a la pre-segmentación. Como se puede apreciar en la Ilustración 5.27 un objeto expresión contiene un atributo *_sentence* (texto de la expresión) y un atributo *_role* (rol de la expresión). Ambos han sido inicializados al realizar la separación en expresiones de una intervención (en la clase Intervention).

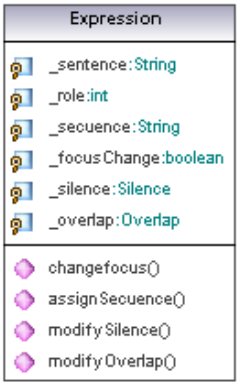

Ilustración 5.27: Clase Expression

En la fase de realización temporal se anotarán los silencios y solapamientos que eventualmente ocurren entre dos expresiones. En concreto, para una expresión determinada, se almacenará información correspondiente al solapamiento que exista con respecto a la expresión anterior (atributo *_overlap*) y el silencio que suceda antes de la expresión posterior (atributo *_silence*). Las características de los atributos *_silence* y

_overlap se podrán modificar a través de los métodos *modifySilence()* y *modifyOverlap()*.

Los atributos *_secuence* y *_focusChange* representan, respectivamente, el tipo de secuencia y el tipo de salto (regular o excepcional, si procede) y ambos serán inicializados en la fase de segmentación a través de los métodos *assignSecuence()* y *changeFocus()*.

CLASE SILENCE

La clase Silence que puede observarse en la Ilustración 5.28 representa un silencio existente entre dos expresiones. Contiene únicamente un atributo *_type* que representa el tipo de silencio y *_duration* que constituye la duración del silencio.

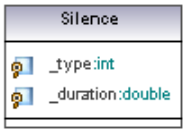

Ilustración 5.28: Clase Silence

Aunque en la ilustración no aparecen métodos relevantes, existen métodos de acceso y modificación para los dos atributos de la clase.

CLASE OVERLAP

La clase Overlap visible en la figura Ilustración 5.29 sigue un formato similar a la clase Silence. Sus atributos representan las características de un solapamiento producido entre dos expresiones. Por un lado, *_units* simboliza el número de palabras que se solapan en caso de que el atributo *_physical* (que representa si un solapamiento es o no físico) contenga un valor verdadero. Por otro lado, *_previousType* y *_nextType* caracterizan el tipo de cambio de tono que realizan los participantes de las dos expresiones involucradas en el solapamiento.

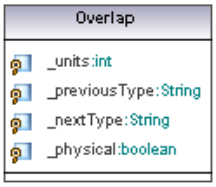

Ilustración 5.29: Clase Overlap

CLASE MICROANALYSISCONTROL

La clase MicroAnalisysControl, que se puede observar en la Ilustración 5.30, maneja la lógica y contiene los algoritmos necesarios para realizar la fase de microanálisis. El atributo *_ctxtVars* almacena toda la información sobre las variables de contexto (nombre, valor y relevancia) por cada expresión analizada, tanto manual como automáticamente.

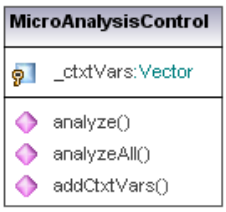

Ilustración 5.30: Clase MicroAnalysisControl

El método *analyze()* realiza una propuesta de acto o actos comunicativos para una expresión. Por otro lado, *analyzeAll()* realiza la misma operación, pero esta vez para todo el conjunto de expresiones de la muestra, comprobando previamente si ha habido algún análisis previo de alguna de las expresiones. Para efectuar estas dos operaciones, es necesario realizar la llamada correspondiente al motor de inferencias que se encuentra en el paquete *Cognos.NL*, (si bien esta integración queda fuera del ámbito del proyecto).

Por último, el método *addCtxtVars()* añade una nueva variable de contexto al array *_ctxtVars*

CLASE SEGMENTATIONCONTROL

La clase SegmentationControl que se visualiza en la Ilustración 5.31 contiene la información y las operaciones de segmentación que se realizan desde la vista View.I_Segmentation. Para ello, tiene un atributo principal que almacena la información y estructura de todos los segmentos creados (*_segments*).

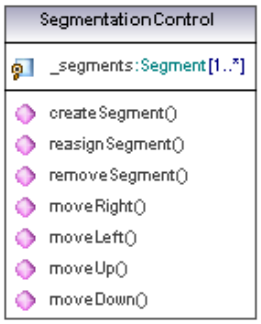

Ilustración 5.31: Clase SegmentationControl

Las operaciones de edición básicas de segmentos se realizan mediante las operaciones *createSegment()* que crea un nuevo segmento, *reasignSegment()* que modifica la estructura de un segmento (esto es, el conjunto de expresiones que contiene), y *removeSegment()* que elimina un segmento del array global.

Dentro del array de segmentos es posible ejecutar operaciones de reordenación de los mismos, para lo cual se han implementado los métodos *moveRight()* y *moveLeft()* que mueven un determinado segmento a la izquierda o a la derecha.

Los métodos *moveUp()* y *moveDown()* permiten realizar las operaciones de modificación de vínculo descomposicional de modo ascendente o descendente para un segmento seleccionado.

CLASE SEGMENT

La clase Segment representa la estructura de información que puede ser editada sobre un segmento a lo largo de la fase homónima. Como se recoge en la Ilustración 5.32, el primer atributo que se ha definido es un array de expresiones (*_expressions*) que corresponde con el conjunto de expresiones contenidas en el segmento. En segundo lugar, se observa el atributo *_intention* (de tipo *String*) que representa la intención del segmento y servirá de identificador. En tercer lugar se ha especificado el atributo *_features*, que contiene la descripción del segmento.

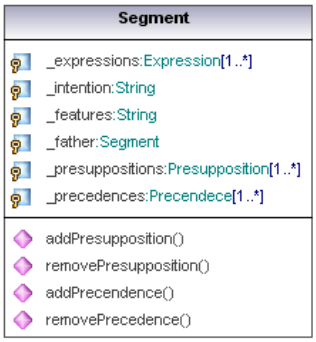

Ilustración 5.32: Clase Segment

Por otro lado, se ha definido un atributo *_father*, que simboliza el objeto Segment correspondiente al segmento padre (en caso de tratarse del segmento base, no tendrá ningún valor).

Los dos últimos atributos de la clase Segment corresponden a dos colecciones de valores: presuposiciones (*_presuppositions*) y vínculos de precedencia (*_precedences*). Para añadir y eliminar presuposiciones y vínculos de precedencia, se han implementado los métodos *addPresupposition()*, *removePresupposition()*, *addPrecedence()* y *removePrecedence()*.

CLASE PRECEDENCE

La clase Precedence, esquematizada en la Ilustración 5.33, contiene la estructura de información necesaria para definir un vínculo de precedencia. *_precedent* y *_cosequent* son dos atributos de tipo String que contendrán la intención del segmento precedente y consecuente, respectivamente. El atributo *_type* definirá el tipo de vínculo de precedencia existente entre los segmentos.

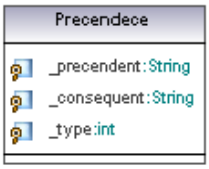

Ilustración 5.33: Clase Precedence

CLASE PRESUPPOSITION

La clase Presupposition realiza una función similar a la clase Precedence pero con respecto a las presuposiciones. En la Ilustración 5.34 se visualiza que únicamente contiene dos atributos: *_name* y *_value*, que representan el nombre y el valor para una presuposición.

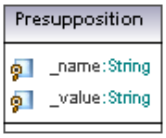

Ilustración 5.34: Clase Presupposition

5.2.3.2 Controller.Utilities

En el paquete Utilities que se muestra en la Ilustración 5.35 se encuentran dos clases (AudioPlayer y FileReader) que recogen algunas funciones auxiliares que solicita el componente GlobalControlAnalysis.

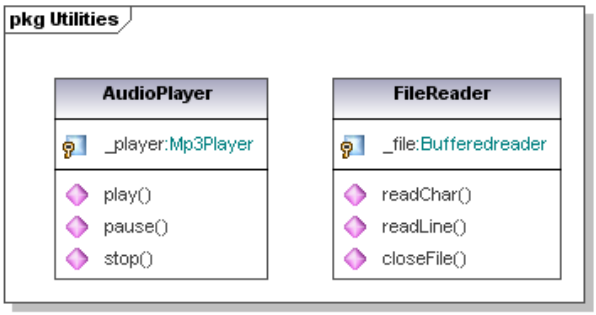

Ilustración 5.35: Utilidades

AudioPlayer es una clase que permite la reproducción de un fichero de audio en formato mp3. Dicha funcionalidad es requerida en la fase de realización temporal y la clase implementada permite reproducir (*play()*), pausar(*pause()*) y parar (*stop()*) la reproducción del fichero .mp3. Dicha clase se apoya en las librerías JLayer proporcionadas por (Javazoom)

FileReader da soporte a la lectura de un fichero externo (útil en la fase de pre-segmentación de la muestra) de modo que contiene métodos para leer un carácter (*readChar()*), una línea (*readLine()*) o cerrar un fichero tras su lectura (*closeFile()*).

5.3 Almacenes de datos

En este apartado se describirá la implementación de la base de datos junto con las soluciones para la semántica no recogida en el diseño conceptual del apartado 4.3. En el esquema relacional que muestra la Ilustración 5.36 se aprecia lo que serán las tablas de las que dispondrá la base de datos Oracle, las claves primarias, claves ajenas y sus correspondientes atributos.

Las tablas que este esquema engloba son CORPUS, SCENE, SAMPLE, INTERVENTION y SENTENCE. Estas entidades (excepto CORPUS) son, en el diseño conceptual, débiles en identificación correlativamente. Esto se ha implementado mediante claves ajenas que forman parte de la clave principal de las entidades débiles, por lo cual la clave primaria de SCENE está compuesta por su clave propia y la clave de identificación de corpus. La clave de SAMPLE está compuesta por su clave propia y la clave de SCENE (es decir, la clave de CORPUS y la de SCENE) y así sucesivamente con INTERVENTION y SENTENCE.

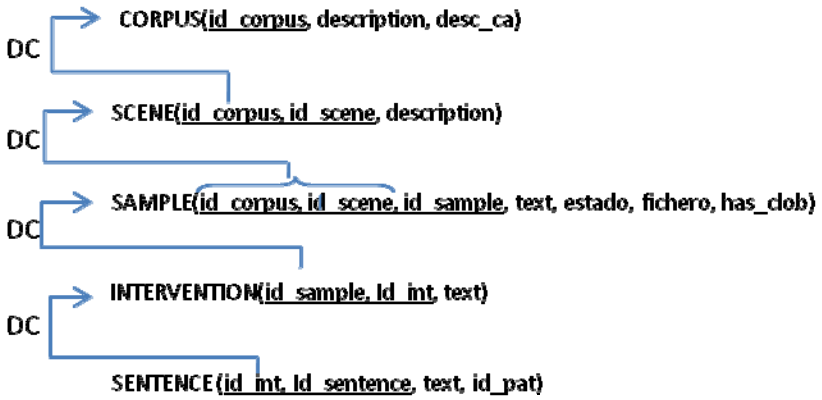

Ilustración 5.36: Grafo Relacional

Para dotar de lógica correlativa a las claves de estas tablas, se creará un disparador para SCENE, SAMPLE, INTERVENTION y SENTENCE en lugar de la secuencia que dispone el resto de tablas. Este disparador calcula la clave de estas tablas en función del valor máximo del último valor de la clave ajena. Esto se ha hecho de esta forma para que la clave de, por ejemplo, SAMPLE tenga un valor correlativo para el corpus 1 (aparezca como 1.1, 1.2, 1.3) y que para el corpus con clave 2 se reinicie la numeración.

Finalmente, es importante señalar que este esquema relacional es solamente una porción de uno global, diseñado de modo común para las aplicaciones Cognos.Dial, *Cognos.NL* y *Cognos.CA*. Concretamente, la parte descrita en este apartado corresponde a la descripción de las tablas que se encuentran involucradas en el proceso de identificación de corpus, escenarios, muestras, intervenciones y frases.

6 Verificación y validación

Este apartado tiene como objetivos realizar la verificación y la validación de la herramienta desarrollada en este proyecto. La verificación tiene como finalidad comprobar que el sistema es correcto y funciona adecuadamente, mientras que la validación va más allá y se encarga de constatar que el sistema desarrollado cumple con los requisitos especificados. Esta fase aumenta la calidad de un desarrollo software y su inclusión es fundamental en todo proyecto.

En el apartado 3.1 se especificaron unas pruebas que, a priori, comprueban todos y cada uno de los requisitos funcionales especificados en el apartado 3.1.2. En este apartado se llevarán a cabo estas pruebas, y si su resultado es válido se podría afirmar que el sistema desarrollado también lo es.

6.1 Ejecución de las pruebas

En este apartado se especificarán los pasos seguidos para efectuar todas las pruebas expuestas en el apartado 3.1.2 partiendo desde la creación de la base de datos y realizando, una a una, las pruebas de validación. Se comentarán los resultados obtenidos en cada una de ellas, y las valoraciones pertinentes.

Para la instalación del sistema completo se requiere tanto la instalación y preparación de una base de datos en un servidor como la instalación del programa editor (ver Anexo III). Tras realizar estos pasos, se ha procedido a ejecutar las pruebas definidas en el apartado 3.3.

6.2 Resultados de las pruebas

Una vez ejecutadas todas las pruebas es necesario realizar un análisis de los resultados obtenidos. Con el fin de no repetir información, se incluye una tabla de resultados que hace referencia a las pruebas definidas en el apartado 3.1.2 y si éstas han sido satisfactorias o no. Si el resultado de una de las pruebas no fuera el deseado, se expondrá la causa de esto y las acciones que se deberán de llevar a cabo para paliar dicha deficiencia.

6.2.1 Tabla de resultados

En la Tabla 2 se aprecian los identificadores de las pruebas seguidos de un icono que indica si éstas han sido satisfactorias (✓) o si, por el contrario, han devuelto un resultado no esperado o si no se han podido realizar (×). Por último se indicará la criticidad de cada una de las pruebas la cual viene determinada por el o los requisitos que verifica cada prueba.

Prueba	Resultado	Criticidad
P1	✓	Alta
P2	×	Baja
P3	✓	Alta
P4	×	Baja
P5	✓	Baja
P6	✓	Media
P7	✓	Alta
P8	✓	Alta
P9	✓	Alta
P10	✓	Alta
P11	✓	Alta
P12	✓	Alta
P13	✓	Alta
P14	✓	Alta
P15	✓	Alta
P16	✓	Alta
P17	✓	Alta
P18	✓	Alta
P19	✓	Alta
P20	✓	Alta
P21	✓	Alta
P22	✓	Media
P23	✓	Alta
P24	✓	Alta
P25	✓	Alta
P26	✓	Alta
P27	✓	Alta
P28	✓	Alta
P29	✓	Alta
P30	✓	Alta
P31	✓	Alta
P32	✓	Alta
P33	✓	Alta
P34	✓	Alta

Tabla 2: Tabla de resultados

Como se puede comprobar, las pruebas P2 y P4 no se han satisfecho. Ambas pruebas se definieron sobre los requisitos R17 y R19 respectivamente y los motivos de estos resultados se detallarán a continuación.

6.2.2 Acciones correctivas

En el apartado anterior (6.2.1) se ha concluido que las pruebas que no se han superado son P2 y P4. En este apartado se explicará el motivo que subyace al incumplimiento de dichas pruebas.

En la matriz de trazabilidad incluida en el apartado 3.3 se puede observar que los requisitos que no se satisfacen son R17 y R19 y tienen como base que se pueda guardar y recuperar un análisis parcial realizado. Desafortunadamente, esta funcionalidad no ha sido implementada totalmente debido a la gran carga de trabajo que conlleva, no obstante es una de las líneas futuras prioritarias y como se puede apreciar en el apartado 7.2.2 será desarrollada a corto plazo.

La no inclusión de esta funcionalidad en el producto final tendrá una mínima trascendencia, pues se trataba de un valor añadido propuesto por el cliente (criticidad baja) y cuya omisión no afectaría a la aprobación y explotación del producto.

7 Conclusiones y líneas futuras

En este último punto se expondrán las conclusiones que se han obtenido tras el período de desarrollo de la herramienta Cognos.Dial y se detallarán las líneas futuras cuya implementación sería deseable para extender tanto la funcionalidad como la potencia de la herramienta. Se analizarán qué objetivos iniciales han quedado cubiertos, cuáles quedan por alcanzar y se propondrán nuevas metas y extensiones de la herramienta para conseguir que la tarea de anotación pragmática de corpus sea en un futuro más sencilla e intuitiva pero a la vez más completa.

7.1 Conclusiones

Para finalizar este proyecto ha de concluirse que se ha satisfecho el principal objetivo que se propuso en el apartado 1.2, es decir desarrollar una única herramienta que englobe todas las fases necesarias para realizar la anotación pragmática de corpus. La aplicación se ha implementado cumpliendo la totalidad de los requisitos de criticidad alta definidos y los pocos que no se han superado las pruebas habían sido definidos como poco críticos. No obstante, se espera que a corto plazo se desarrolle una nueva versión que consiga cumplir con todos estos requisitos e incluya alguna de las líneas futuras que después se propondrán.

Antes de la realización de este proyecto no existía ninguna herramienta similar que englobase todas las fases de anotación de corpus. Por lo tanto, este trabajo constituye una aportación al ámbito de la interacción natural. Además el trabajo realizado ha dado lugar a un proyecto de investigación en el grupo LABDA en el cual se está trabajando actualmente para mejorar la herramienta y ampliar su funcionalidad.

En el ámbito personal, el conocimiento adquirido durante la carrera por la autora de este proyecto ha sido de vital importancia para desarrollar la herramienta. No obstante, al ser el ámbito de actuación de la misma en área de la lingüística, este conocimiento ha tenido que ser ampliado para la comprensión de términos y procesos de este ámbito. Además ha de mencionarse que el desarrollo de la herramienta ha mejorado notablemente el nivel de programación de interfaces gráficas en Java del equipo de desarrollo.

7.2 Líneas futuras

A continuación se describirán las líneas futuras de la herramienta Cognos.Dial. Una vez que fueron implementadas las cuatro primeras fases del análisis en las que se centra la herramienta (pre-segmentación, realización temporal, microanálisis y segmentación), el proyecto dio lugar a un trabajo de investigación en el cual se completaron las demás fases del análisis individual de muestras de diálogo. De este modo, el trabajo ya implementado pero que queda fuera del ámbito del proyecto se describe en el apartado de líneas futuras ya implementadas (7.2.1). El resto de líneas futuras se encuentra subdividido según la prioridad de su función en líneas futuras a corto y a largo plazo.

7.2.1 Líneas futuras ya implementadas

Se ha incluido una nueva fase de análisis que gestiona el compromiso. En esta fase se pueden añadir los eventos que afectan al compromiso del diálogo (aumentándolo o disminuyéndolo), así como las técnicas que se aplican para aumentar el grado de compromiso.

Tras el análisis del compromiso, se ha añadido otra nueva fase denominada “Operativa” en la cual se anotan las tareas que se llevan a cabo en el desarrollo del diálogo. Además, en esta fase se recogen las entradas, salidas y efectos de cada tarea anotada.

La última fase del análisis que se encuentra actualmente implementada es “Estructural”. En esta fase se generan los diagramas de estados a partir de la información anotada para cada uno de los segmentos.

Por último, Cognos.Dial ha sido integrada con las aplicaciones *Cognos.NL* y *Cognos.CA*, que son, respectivamente una herramienta para el análisis del lenguaje natural y un editor de actos comunicativos

7.2.2 Líneas futuras que se implementarán a corto plazo

Próximamente estará completa tanto la exportación como la importación de todo el análisis individual a un fichero XML. También será posible subir a una base de datos de muestras individuales toda la información de análisis realizado al finalizar éste, con el fin de que sean sometidos a un análisis global.

Se permitirá la edición con carácter retroactivo, es decir se podrá volver a anotar una fase anterior tras haberla completado y haber pasado a una fase posterior. De este modo, también habrá que implementar los chequeos necesarios para asegurar la corrección del análisis.

Se traducirán todas las etiquetas a otros idiomas de modo que la aplicación sea multilingüe. De momento, la herramienta se encuentra en inglés y próximamente se introducirá el español, no obstante se hará extensible a otros idiomas en un futuro.

Se creará un fichero de configuración con las siguientes posibilidades:

- Opción de escoger las opciones de análisis: según las diferentes definiciones de discurso.
- Elegir el color de los dos roles y el nombre de los roles.
- Cambiar el directorio por defecto a la hora de cargar una muestra desde un fichero de texto.
- Arrancar la herramienta *Cognos.NL* bloqueando o no la pantalla de Cognos.Dial
- En la realización temporal, incluir la posibilidad de elegir los colores de cada tipo de silencio.
- Incluir los datos de los usuarios (nombres de usuario y claves cifradas).
- Modificar el número de corpus que se muestran en la lista desplegable y según el usuario dar la posibilidad de ordenarlos alfabéticamente o frecuentemente.
- Configurar el número de cambios a deshacer.

En la fase de Segmentación, permitir reutilizar una intención ya existente en el corpus para evitar que se creen segmentos redundantes. Además en esta fase es necesario chequear la validez de los vínculos de precedencia que introduce el usuario con el objetivo de evitar bucles

También se propone ampliar la funcionalidad de la fase Operativa; incluyendo en la base de datos los ámbitos y las funciones, permitir cortar y pegar tareas de una expresión a otra, así como permitir el borrado de tareas.

Otras acciones menores a afrontar en el futuro se enumeran a continuación:

- Completar la gestión del botón cerrar muestra existente en la barra de herramientas para que avise en todas las fases si se desea guardar la muestra. Incluir otro botón en la barra que de la opción de borrar una muestra.
- Ampliar los casos en los que se muestra texto informativo en la barra de información inferior.
- Añadir una interfaz que permita al usuario autenticarse para conectarse y desconectarse de la base de datos.
- Incluir una barra de menú superior que recoja toda la funcionalidad de la herramienta de modo que exista siempre una alternativa al realizar una operación.

- En la barra de menú, en el selector de corpus, mostrar los diez últimos corpus que se han cargado y añadir un ítem en la lista desplegable "ver todos".
- En la fase de presegmentación, si el chequeo no es correcto porque falta texto por asignar rol, resaltar en algún color llamativo cuáles son las porciones de texto que faltan por asignar.
- Adaptar las dimensiones de la interfaz de usuario de modo que permita cualquier tipo de resolución, ya que actualmente se encuentra limitada a 1280x1024 pixeles.
- Dar funcionalidad al botón de ayuda para que muestre información acerca del manejo de la herramienta.

Finalmente, será necesario implementar las funciones pertinentes en el análisis global de un corpus. Dicho análisis consta de las siguientes fases: gestión de proyecto, unificación de tareas, unificación de segmentos, gestión de excepciones de ámbito atencional y aprendizaje del compromiso. Sería además deseable que los resultados de esta fase se trasladaran automáticamente a las bases de conocimiento de un modelo de diálogo.

7.2.3 Líneas futuras que se implementarán a largo plazo

Destaca la posibilidad de realizar la importación y exportación a XML de un análisis incompleto. De este modo sería posible mostrar en la pestaña de codificación el código XML parcial según las fases que se hayan completado. Esto también permitiría en el futuro deshacer los cambios que se han realizado en cualquier fase del análisis.

Otra línea interesante sería integrar un reconocedor de voz y aumentar la integración con Cognos.NL, para así en la fase de microanálisis dividir las expresiones automáticamente en el momento en el que una muestra encaja con dos patrones.

Finalmente, sería deseable incluir una interfaz de usuario que permita editar el fichero de configuración para aumentar la flexibilidad y las posibilidades de la interfaz gráfica de la herramienta.

8 Bibliografía

Altova UModel. (s.f.). Obtenido de http://www.altova.com/umodel.html

Austin, J. L. (1962). *How to Do Things With Words.* Oxford University press.

B. Grosz, C. S. (1986). Attention, intention and the structure of discourse.

Calle, J. (2004). *Interacción Natural mediante procesamiento intencional: Modelo de Hilos en diálogos. Tesis Doctoral.* Madrid.

Clark, H. H. (1996). Using language. . *Cambridge University Press.*

Collabnet Subversion. (s.f.). Obtenido de http://www.collab.net/

D. Cuadra, M. C. (2009). Anotación Pragmática de Diálogos en XML.

D. Jurafsky, E. S. (1997). *Switchboard-DAMSL Labeling Project Coder's Manual.*

D. Scott, D. K. (1997). *Discourse Modeling. Survey of the state of the art in Human Language Technology.*

DAMSL. (s.f.). Obtenido de http://www.cs.rochester.edu/research/speech/damsl/RevisedManual/

Hobbs, J. R. (1985). On the coherence and structure of discourse.

J. A. Levin, J. A. (1977). *Dialogue games: Metacomunications strategies for natural language interaction.*

Javazoom. (s.f.). Obtenido de http://www.javazoom.net/index.shtml

LaBDA. (s.f.). Obtenido de http://basesdatos.uc3m.es

McKeown, K. R. (1985). *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.*

McTear, M., Jokinen, K., & Larson, J. (2008). Evaluating new methods and models for advanced speech-based interactive systems.

NetBeans 6.7. (s.f.). Obtenido de http://www.netbeans.org/

P. R. Cohen, M. J. (1997). *QuickSet: Multimodal interaction for distributed applications, in the Proceedings of the Fifth International Multimedia Conference.*

R. Cattoni, M. D. (2001). Building a corpus of annotated dialogues: the ADAM experience .

R. Cooper, S. L. (1998). Dialogue Moves and Information States.

The Institute of Electrical and Electronics Engineers. (1998). *IEEE Recommended Practice for Software Requirements Specifications.*

Transcriber. (s.f.). Obtenido de http://trans.sourceforge.net/en/presentation.php

W. C. Mann, S. A. (1987). Rhetorical structure theory: a theory of text organization.

Wittgestein, L. (1988). *Investigaciones Filosóficas.*

Anexo I: Glosario

Base de conocimiento: Almacén en el que se guardan ciertos datos que componen un conocimiento específico.

COGNOS KB: Base de conocimiento integrada del sistema COGNOS.

COGNOS Toolkit: Conjunto de herramientas de adquisición y gestión de conocimiento para la interacción natural.

Corpus: Conjunto lo más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación.

Diagrama o Modelo Entidad-relación: Herramienta para el modelado de datos de un sistema de información.

Discurso: Porción de intervención dirigida a un solo fin o intención, desarrollada por un solo participante y sin que haya otra intervención anidada en medio.

Escenario: Parte de las muestras de un corpus que tratan el mismo tema o guardan alguna relación entre ellas.

Evento: Acontecimiento que sucede durante el transcurso de un diálogo.

Expresión: Porción de intervención que no contiene ningún signo de punto y aparte ni ningún carácter de retorno de carro.

Feedback (realimentación): En el ámbito de la ingeniería de software, este concepto se refiere a la acción de obtener opiniones y sugerencias de parte de terceros sobre una herramienta.

Inferencia: Acción y efecto de sacar una consecuencia o deducir algo de otra cosa.

Intención: Fin que persigue la realización de un segmento.

Interfaz de usuario: Mecanismo de comunicación usual entre los programas informáticos y sus usuarios.

Intervención: Ejecución de un turno por parte de un participante o secuencia consecutiva de expresiones dirigida a uno o a varios fines.

Juego de diálogo: Intercambio de acciones comunicativas.

Latencia: suma de retardos temporales dentro de una red.

Librería de programación: conjunto de funciones externas que amplían las funciones propias del lenguaje de programación.

Línea de discurso: Sinónimo de expresión: porción de intervención que no contiene ningún signo de punto y aparte ni ningún carácter de retorno de carro.

Máquina virtual de java: programa ejecutable en una plataforma específica capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial generado por el compilador del lenguaje Java.

Microanálisis: Fase de anotación en la que se asigna un conjunto de actos comunicativos a cada una de las expresiones del diálogo.

Motor de inferencia: programa que, a partir de una carga de datos previos, es capaz de realizar inferencias sobre estos datos.

Muestra: Texto que contiene un diálogo entre dos individuos.

Patrón: Elemento principal en una gramática relajada.

Patrón de diseño: Diseño que busca solución a problemas de programación comunes.

Pragmática: Disciplina que estudia el lenguaje en su relación con los usuarios y las circunstancias de la comunicación.

Precedencia: Relación existente entre dos segmentos (comúnmente hermanos) que indica la necesidad de inicio o finalización de uno para que suceda el inicio o la finalización del otro.

Secuencia: Estructura lingüística asociada a una expresión.

Segmento: Instancia que tiene como fin hacer progresar una intención con un resultado que puede ser satisfactorio o insatisfactorio.

Segmento base: Segmento principal que abarca todas las intervenciones de un diálogo.

Silencio: Abstención de hablar que sucede entre dos expresiones, ya sean éstas del mismo participante o no.

Sistema gestor de base de datos: Tipo de software muy específico dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Solapamiento: Evento que sucede cuando dos expresiones (de diferente rol) coinciden a la vez en el mismo intervalo de tiempo.

Tarea: Acción que realiza un participante del diálogo externa a la conversación y necesaria para seguir su transcurso.

Técnica: Acción que realiza uno de los participantes del diálogo para realizar un cambio (normalmente positivo) en el compromiso en la conversación.

Token: Elemento de una gramática relajada que identifica a los terminales de esta. Está compuesto a su vez de diferentes elementos.

Toolkit: Conjunto de aplicaciones de un mismo sistema integrado.

Ventana emergente (pop-up): Tipo de interfaz de usuario que se activa generalmente sin que el usuario lo solicite cuando se necesita cierta información antes de realizar o continuar una acción.

Vínculo descomposicional: Vínculo que se crea en el momento en el que un segmento se descompone en uno o varios subsegmentos.

Anexo II: Acrónimos

API: ***A****pplication* ***P****rogramming* ***I****nterface*. Conjunto de funciones que proporciona un cierto lenguaje de programación al desarrollador.

HTML: ***H****yper* ***T****ext* ***M****arkup* ***L****anguage.* Lenguaje de marcado predominante en páginas Web.

MVC: ***M****odel-****V****iew-****C****ontroller*. Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

PL/SQL: ***P****rocedural* ***L****anguage/****S****tructured* ***Q****uery* ***L****anguage***.** Lenguaje de programación incrustado en Oracle y PostgreSQL.

SGBD: ***S****istema de* ***G****estión de* ***B****ase de* ***D****atos.* Tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

XML: *E****x****tensible* ***M****arkup* ***L****anguage.* Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

Anexo III: Manual de Instalación

La instalación de la herramienta de anotación pragmática requiere, en primer lugar, tener copiada la carpeta Cognos_Dial en el disco duro de la máquina del cliente (por ejemplo, en el directorio C:\). Puesto que Cognos.Dial ha sido desarrollada en Java, es necesario tener su Entorno de Ejecución instalado (JRE). Por último, la máquina cliente ha de tener conexión a la red VPN de la Universidad Carlos III de Madrid ya que base de datos a la que se conecta el sistema se encuentra actualmente en un servidor del grupo LaBDA.

A continuación se explicarán los pasos necesarios para instalar tanto el Entorno de Ejecución de Java como la Red Privada Virtual

Instalación del JRE

El Entorno de Ejecución de Java es posible descargarlo gratuitamente desde la siguiente página web: http://java.sun.com/javase/downloads/index.jsp. Únicamente es necesario descargar el archivo correspondiente al enlace *Java Runtime Environment (JRE) 6* (Ilustración AIII-1) y seguir los pasos para su instalación.

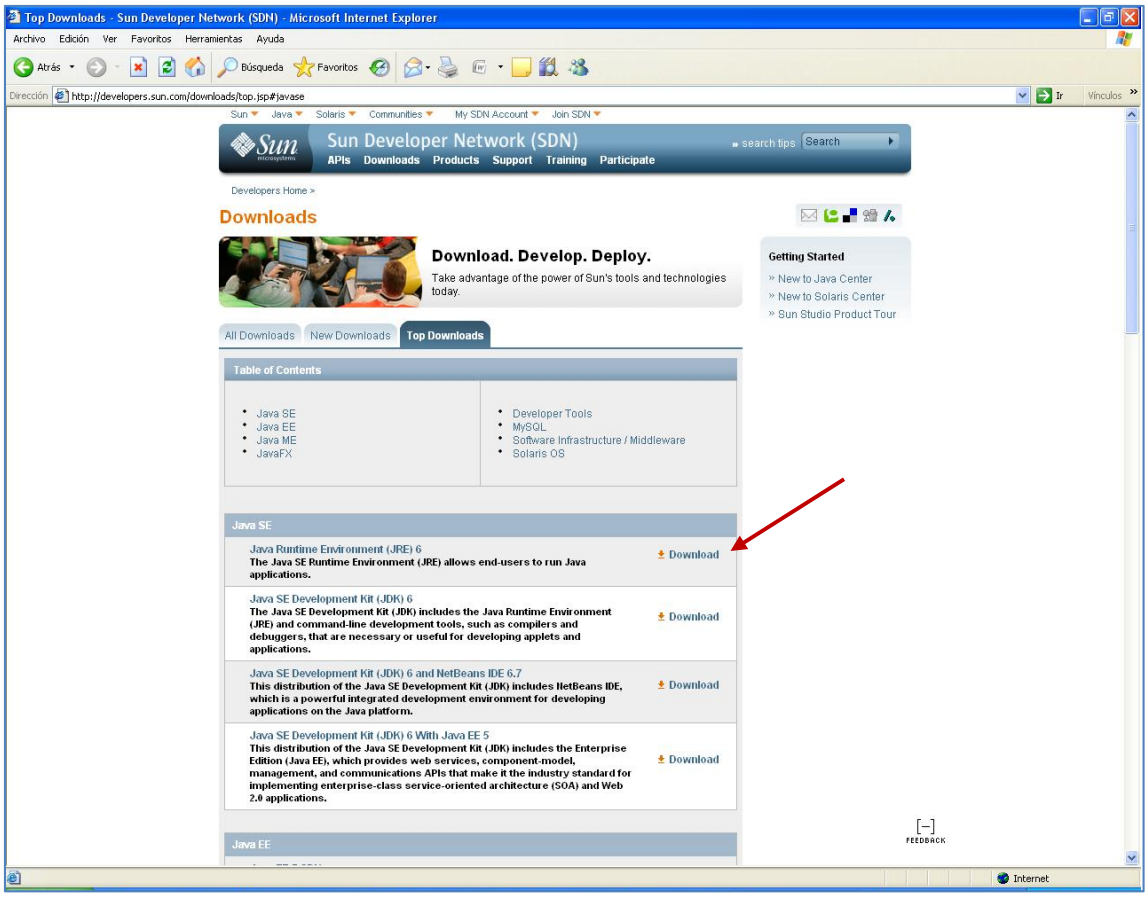

Ilustración AIII-1: Descarga JRE

Configuración de la VPN

Para configurar la VPN en el sistema operativo y poder acceder a la Red Privada Virtual de la Universidad Carlos III, basta con seguir los pasos que se explican en el siguiente tutorial on-line: https://asyc.uc3m.es/index.php?Id=168 (Ilustración AIII-2).

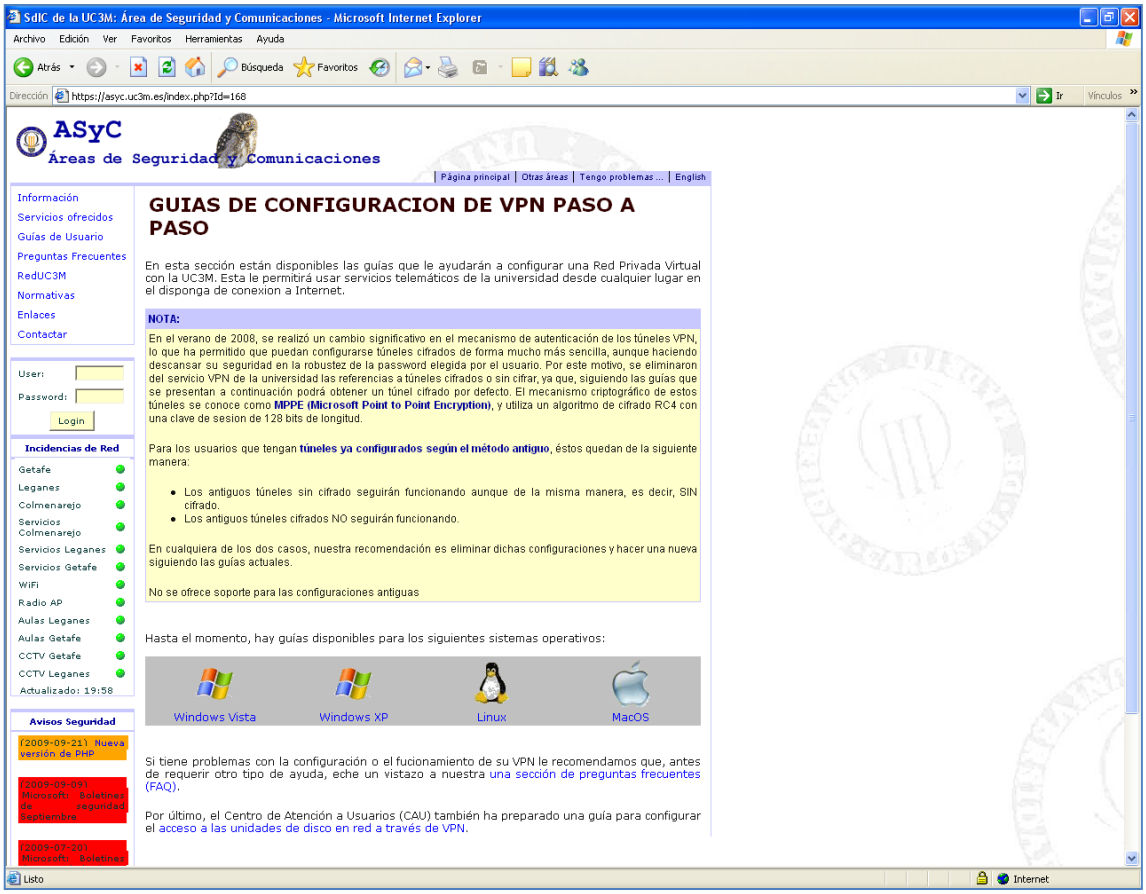

Ilustración AIII-2: Configuración de la VPN

Una vez realizados estos pasos, para ejecutar la aplicación únicamente será necesario hacer doble click sobre el archivo cognos_dial.jar contenido en la carpeta Cognos_Dial.

Anexo IV: Manual de usuario

A lo largo de esta sección se explicará el manejo de la herramienta Cognos.Dial para cada una de las fases del análisis y anotación de corpus lingüístico. La aplicación consta de ocho pestañas y una barra de herramientas común para todas ellas.

La primera pestaña (codificación) permite al usuario visualizar un código intermedio utilizado por la aplicación para guardar análisis parciales de corpus. Dicho código podrá ser modificado por el usuario, siendo éste responsable de cualquier efecto no deseado que pueda ocasionar.

Las siete pestañas siguientes corresponden a cada una de las fases de análisis y anotación de corpus y su funcionamiento se detallará individualmente en los siguientes apartados.

Barra de herramientas general

La barra de herramientas mostrada en la Ilustración AIV-1 aparece en la parte superior de la ventana principal de la aplicación y el usuario puede arrastrarla a cualquier otra parte de la pantalla mediante la técnica *drag and drop*.

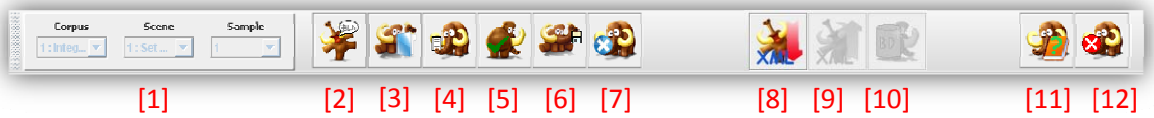

Ilustración AIV-1: Barra de herramientas general

A la izquierda de la barra [1] aparecen tres *combo-box* de selección que permiten la identificación de la muestra sobre la que se desea trabajar. En primer lugar el usuario debe elegir el corpus, después el escenario y finalmente la muestra. Una vez realizada la identificación, se debe pulsar el botón que permite la recuperación de la muestra desde la base de datos [3].

En ese momento se activará únicamente la pestaña de pre-segmentación si la muestra no ha sido analizada previamente. En caso de que ya se haya realizado un análisis total o parcial de la muestra, se activarán las pestañas correspondientes, mostrándose el análisis realizado en cada una de ellas.

Para crear un nuevo corpus basta con pulsar el botón de *nuevo* [4] y en ese momento se mostrará un pop-up en el que se podrá escribir el nombre del corpus. Seguidamente aparecerá otro pop-up para introducir el nombre de un escenario. Una vez introducida la información del corpus y el escenario, aparecerá otra ventana que da la opción de crear una muestra vacía o adquirir el texto del diálogo de un fichero existente en el disco duro. Cualquiera de las dos opciones elegidas activa la pestaña de pre-

segmentación, bien con el texto del fichero o bien en blanco para que el usuario escriba directamente el texto de la muestra.

El botón de *descripción* [2], que aparece en primer lugar dentro de la barra de herramientas, muestra una ventana con la descripción del corpus y del escenario sobre el cual se está trabajando.

El botón de *chequeo* [5] estará activado en aquellas pestañas que requieran la revisión de una serie de condiciones de validez que tiene que cumplir el análisis realizado por el usuario. Dichas pestañas son las correspondientes a la fase de pre-segmentación, microanálisis y segmentación. En cualquiera de ellas, si el chequeo da un resultado negativo, no se podrá acceder a las siguientes pestañas del análisis y permanecerán desactivadas hasta que el usuario revise y solucione los errores.

Mediante el botón *guardar* [6], se podrá guardar el análisis realizado sobre una muestra en cualquiera de las fases que se encuentre.

El botón *cerrar muestra* [7] permite al usuario cerrar la muestra actual sobre la que está trabajando y abrir otra muestra o crear una nueva. Al pulsar este botón se le mostrará una advertencia al usuario si el análisis no ha sido guardado antes de cerrar, permitiéndole hacerlo en ese momento o desechar el trabajo realizado.

Los botones [8] y [9] permiten, respectivamente, *importar* o *exportar* un archivo en XML para recuperar o almacenar un análisis en dicho formato.

El botón de *upload* [10] almacena en la base de datos el análisis completo realizado. Este botón sólo permanecerá activo al terminarse completamente el análisis operativo.

Fase I: Pre-segmentación

En la ilustración AIV-2 se muestra una muestra de un diálogo entre dos personas que ha podido ser escrito directamente por el usuario en el panel de texto o recuperado de un fichero del disco duro mediante el botón [1].

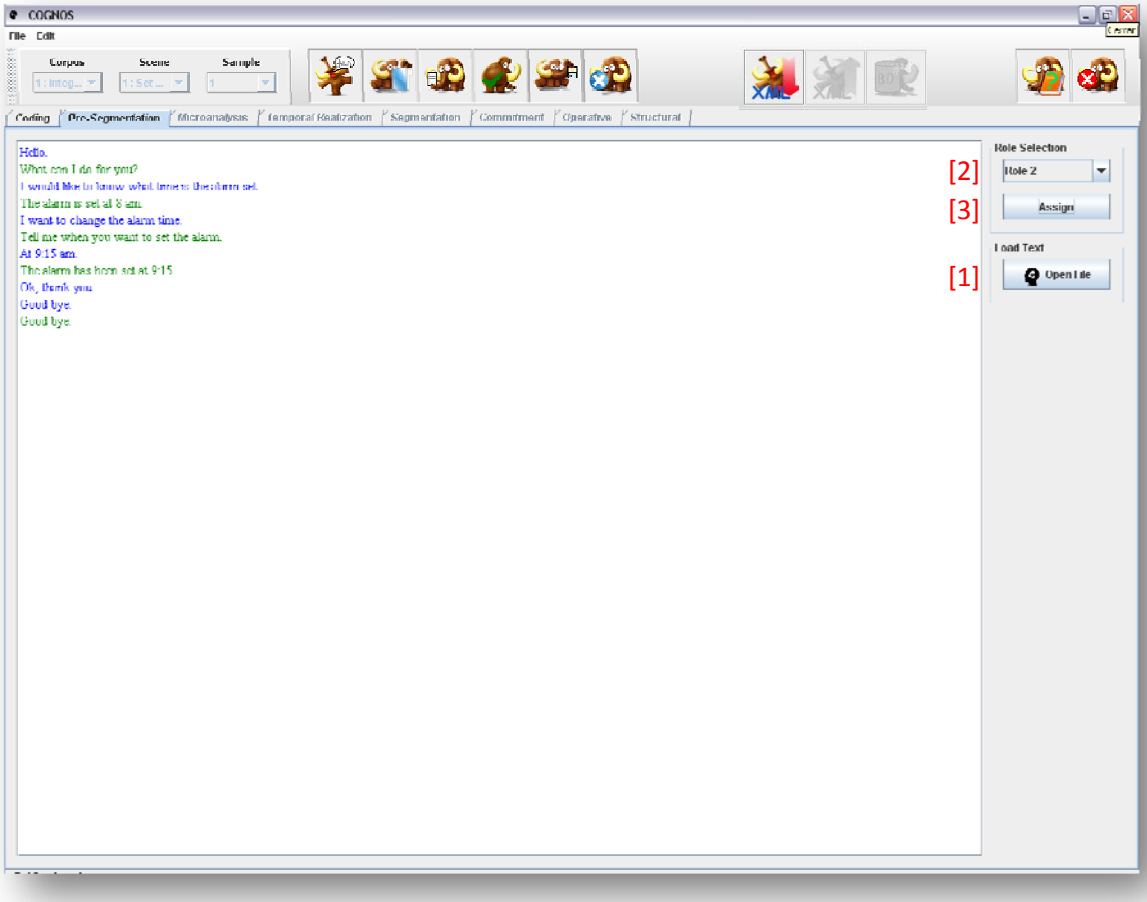

Ilustración AIV-2: Pre-Segmentación

Para asignar un rol determinado a cada fragmento del texto, primero se debe seleccionar con el ratón dicho fragmento y seguidamente, el usuario debe seleccionar en el combo-box [2] el rol que desea asignarle. Finalmente se pulsa el botón *asignar* [3] y en ese momento el texto seleccionado cambia de color (azul o verde) según el rol elegido. La letra del texto permanece de color negro mientras no se le haya asignado ningún rol.

Una vez analizado todo el texto, se guarda el análisis mediante el botón *guardar* de la barra de herramientas y se activan las pestañas de microanálisis, realización temporal y segmentación. Esta acción se realizará satisfactoriamente siempre y cuando no hayan quedado fragmentos de texto sin analizar. Dicha comprobación puede ser realizada independientemente del guardado pulsando el botón *chequear* de la barra de herramientas general.

Fase II: Microanálisis

En esta fase del análisis del diálogo, cuya interfaz gráfica se muestra en la Ilustración AIV-3 se asignan actos comunicativos editados mediante *Cognos.CA* a cada intervención. A su vez se editarán variables de contexto existentes en el diálogo.

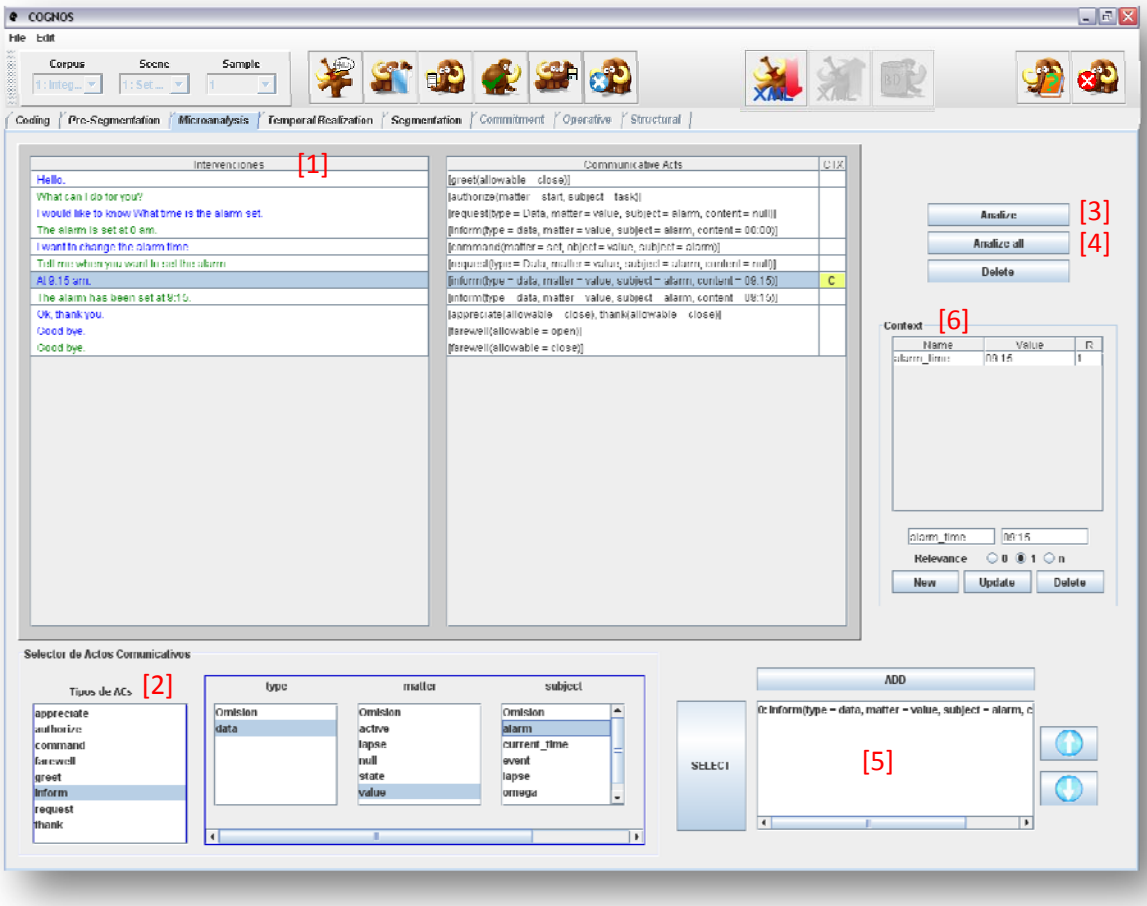

Ilustración AIV-3: Microanálisis

El análisis de esta fase puede ser completamente automático si la base de conocimiento dispone de suficiente información, para ello se permite analizar una expresión [3] o analizar todas [4].

En primer lugar, para realizar la asignación de un acto comunicativo se ha de seleccionar tanto la intervención correspondiente como el acto comunicativo adecuado. Para ello, en la tabla de intervenciones [1] será necesario seleccionar la intervención y, en el selector de actos comunicativos [2], se identificará el que corresponda con esa expresión. Tras la selección de un acto comunicativo se usa el botón *select* para añadirlo a la lista de gestión de conjunto [5] donde se podrá editar el orden de los actos comunicativos dentro de la asignación. Una vez terminada la edición del conjunto que se desea asignar, mediante el botón *ADD* se asignarán los actos comunicativos seleccionados a la intervención deseada.

Si la intervención que se está editando contiene variables de contexto, estas se gestionarán a través de la edición context [6] donde se permite editar las variables que aparecen en la intervención insertando su nombre, valor y relevancia, la cual puede ser 0, 1 o n. las intervenciones que contengan variables de contexto se identifican mediante un cuadro amarillo con una C.

Fase III: Realización temporal

En esta fase del análisis, mostrada en la Ilustración AIV-4, aparecen las intervenciones de forma escalonada dentro de un panel. Esta disposición facilita la visualización de silencios y solapamientos entre intervenciones. El color de fondo de cada una de las intervenciones corresponde al de los roles asignados en la fase de pre-segmentación.

Mediante unos controles de audio incluidos en la pestaña, el usuario puede abrir [3] el archivo de audio correspondiente al diálogo de la muestra y almacenado en el disco duro del ordenador para reproducirlo [1]. De esta forma se permitirá realizar cómodamente la notación de las silencios y solapamientos del diálogo. Para pausar la reproducción se volverá a pulsar el botón [1] y para pararla el botón [2].

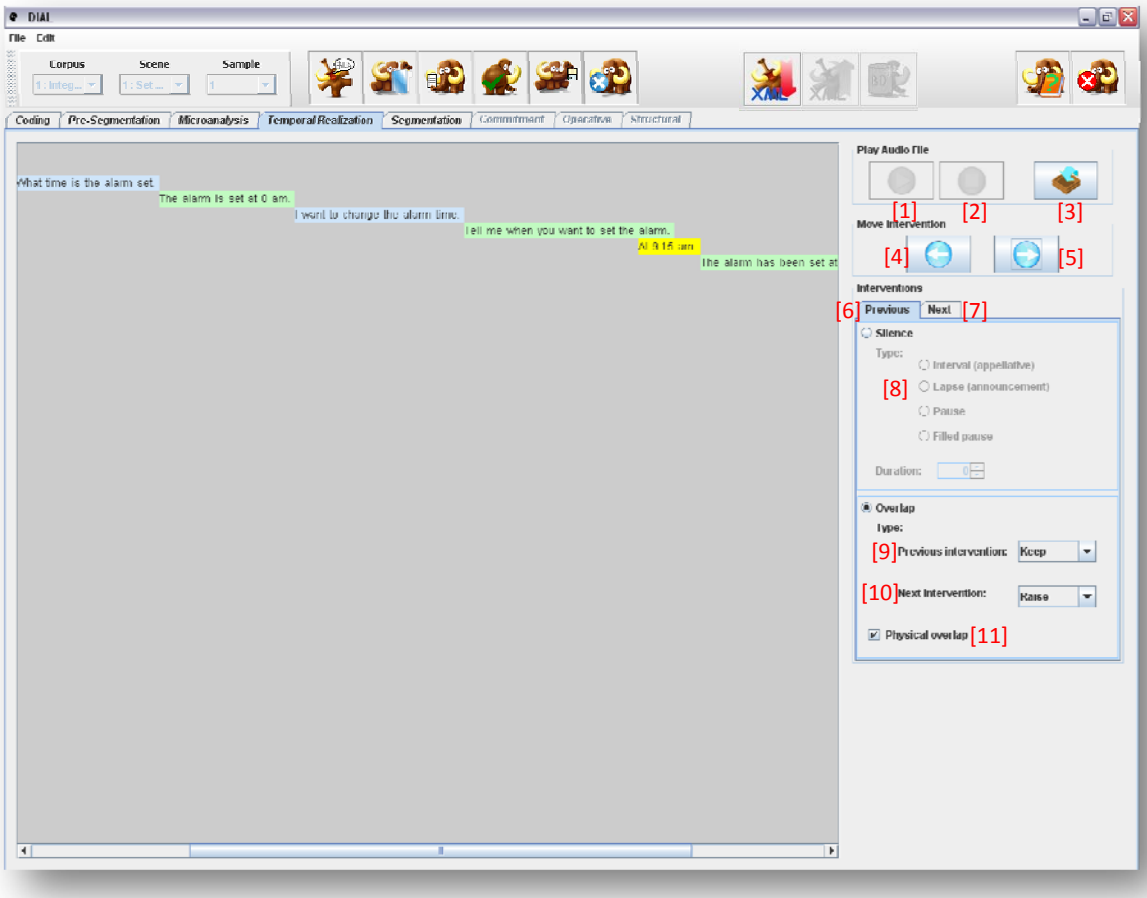

Ilustración AIV-4: Realización temporal

Para introducir un evento temporal, se debe seleccionar previamente una intervención (en la ilustración 8: “At 9:15 am.”). Mediante los botones *izquierda* [4] y *derecha* [5] se moverá la intervención para realizar solapamientos o silencios con respecto a la intervención anterior.

Al mover una intervención a la izquierda puede ocurrir lo descrito en una de las siguientes opciones:

- Si la intervención no tiene ni silencios ni solapamientos con respecto a la intervención de la izquierda, la intervención se solapará una palabra con la anterior por cada vez que se pulse el botón *izquierda*.
- Si la intervención ya se encuentra solapada con la anterior, se solapa una palabra más con ella, pudiéndose solapar completamente.
- Si existe un silencio entre la intervención seleccionada y la anterior, dicho silencio se reduce una unidad (0,5 segundos).

Al mover una intervención a la derecha puede ocurrir lo siguiente:

- Si la intervención seleccionada no tiene ni silencios ni solapamientos con respecto a la intervención de la izquierda, se creará un silencio de una unidad (0,5 segundos) con respecto a la intervención anterior y se aumentará una unidad por cada vez que se pulse el botón *derecha*.
- Si la intervención se encuentra solapada con la anterior, el solapamiento se reduce una palabra por cada vez que se pulse el botón *derecha* pudiéndose eliminar el solapamiento completamente.
- Si ya existe un silencio entre la intervención seleccionada y la anterior, dicho silencio se aumenta una unidad (0,5 segundos) por cada vez que se pulse el botón *derecha*.

Mediante el panel *intervenciones* que posee dos pestañas: *previa* [6] y *siguiente* [7] se manejarán los eventos temporales que suceden a la izquierda o derecha de la intervención seleccionada.

En dichos paneles se puede seleccionar el *tipo* de silencio [8]: intervalo, lapso, pausa o pausa oralizada. Cuando se seleccione el tipo, la marca de silencio cambiará de color y se mostrará la inicial del tipo seleccionado.

El solapamiento entre dos intervenciones puede no ser físico, en cuyo caso no se encontrará activado el *checkbox* de solapamiento físico [11]. No obstante, para todos los solapamientos, sean o no físicos, se permitirá elegir el tipo de cambio de tono que han realizado los hablantes. Para la intervención solapada o previa [9] el hablante puede realizar una de estas acciones: bajar, mantener, elevar el tono o abortar la intervención. Para la intervención que solapa o siguiente [10], el hablante puede bajar, mantener, elevar el tono o robarle el turno al otro hablante.

Fase IV: Segmentación

En la pestaña de segmentación que se muestra en la Ilustración AIV-5 aparece una lista con todas las intervenciones [2] del diálogo en la cual cada una de ellas mantiene el color de su rol. A la izquierda de las intervenciones se visualizarán los segmentos [1] que vaya creando el usuario. Al comienzo de la fase únicamente aparece el segmento base, el cual es obligatorio en todos los análisis. Dicho segmento abarca todo el diálogo y no es editable en ningún caso.

La herramienta permite visualizar los actos comunicativos en vez de las intervenciones, cuando se presione el botón *Mostrar ACs* [4]. En ese momento dejarán de mostrarse las intervenciones, en su lugar se mostrarán los actos comunicativos y el botón [4] mostrará el texto: *Mostrar Intervenciones.*

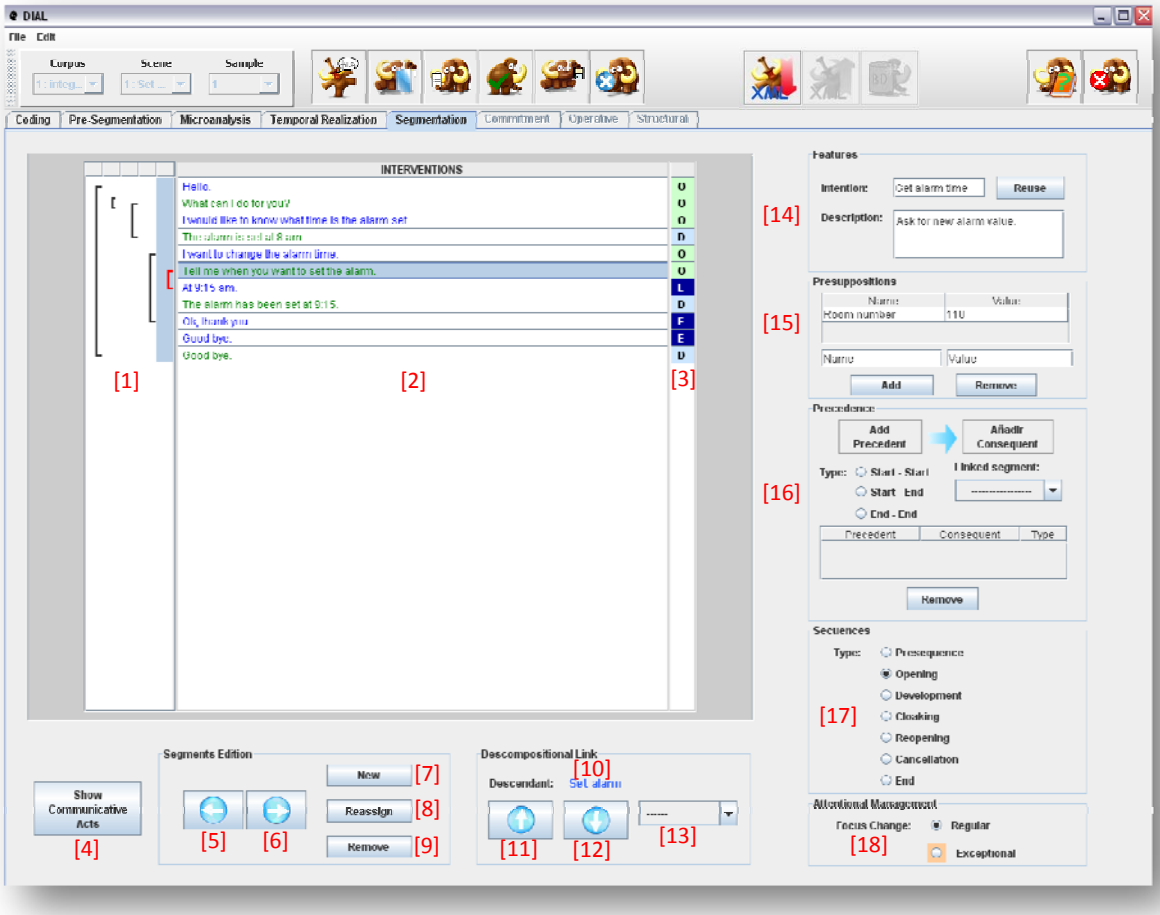

Ilustración AIV-5: Segmentación

Para crear un nuevo segmento, el usuario debe seleccionar un conjunto de intervenciones o ACs. En ese momento, se pulsa el botón *Nuevo* [7] y aparece una nueva columna con el nuevo segmento creado. A dicho segmento es obligatorio asignarle una *intención* (siendo posible reutilizar una existente del mismo corpus al que pertenece la muestra) y una *descripción* de la misma [14].

Dentro de la edición de segmentos, es posible mover uno a la *izquierda* [5] o *derecha* [6], cambiar el conjunto de intervenciones o ACs de un segmento con el botón *reasignar* [8] o *eliminar* [9] un segmento seleccionado.

Cualquier segmento menos el base es descendiente de otro segmento. Al seleccionar cualquier segmento se mostrará la intención de su antecesor o padre [10]. Se permite cambiar el vínculo descomposicional de un segmento, asociándole un padre diferente al actual. De esta forma, el segmento puede *subir* [11] en la jerarquía de segmentos, en cuyo caso su nuevo padre pasa a ser el padre de su antiguo padre o *bajar* [12] en la jerarquía. Para bajar en la jerarquía hay que seleccionar un segmento hermano en el combo-box [13]. El hermano seleccionado pasará a ser su nuevo padre.

Un segmento puede tener presuposiciones [15], para añadir una es necesario insertar su nombre y su valor. Además es posible eliminar presuposiciones creadas previamente para un segmento.

Es posible añadir vínculos de precedencia [16] entre segmentos. Para añadir uno, es necesario seleccionar un segmento, seguidamente elegir un tipo de precedencia (inicio-inicio, inicio-fin o fin-fin) y elegir el segmento vinculado. En ese momento se activan los botones *añadir precedente* y *añadir consecuente*. Cuando se pulse uno de ellos se añadirá la precedencia a la tabla de precedencias del segmento. Será posible eliminar las precedencias mediante el botón *eliminar*.

Cada intervención tiene asociada una secuencia de la cual se podrá editar su tipo [17]: presecuencia (P), apertura (O), desarrollo (D), oscurecimiento/abandono (C), reapertura (R), cancelación (X) y cierre (E). A la derecha de las intervenciones aparece una columna [3] en la cual se puede visualizar el tipo de secuencia para cada intervención.

Por último, esta pestaña permite editar la gestión atencional de las secuencias de tipo pre-secuencia, apertura y reapertura. Se podrá determinar la existencia de saltos excepcionales (la intervención aparecerá con fondo naranja), siendo éstos regulares por defecto.