Universidad “José Martí”
Facultad de Informática
Carrera de Ingeniería Informática

Trabajo de Diploma para optar por el título de Ingeniería en Informática

Título: Sistema Multiagente para revertir el miedo presente en los estudiantes durante su interacción con un ambiente de enseñanza-aprendizaje virtual

Autor: Yoelbis Bonet Gallardo
Tutora: MSc. Irina Machado Mutis

Sancti Spíritus, Cuba
Curso 2015- 2016

Pensamiento

Aprendí que el coraje no es la ausencia del miedo, sino el triunfo sobre él. El hombre valiente no es aquel que no siente miedo, sino el que conquista ese miedo.

Nelson Mandela

Agradecimientos

A mis padres por todo su apoyo y comprensión.
A mi hermano por su apoyo incondicional siempre.

A mis amistades que siempre me dieron la fuerza para seguir adelante.

A Maday y Lisandra por su fuerza, energía y ganas siempre de hacer las cosas cada vez mejor.

A Betty, Ismary, Lisandrita, Humberto, Denis, Daniel y Edel, las palabras sobran.

A mis compañeros por regalarme 5 maravillosos años de mi vida.

A Irina por su paciencia y dedicación, y sobre todo, por siempre confiar en mí.

A Lydia por ser un ser humano tan maravilloso.

A Lisa Wright por la fuerza e influencia que siempre han tenido sus palabras sobre mí.

A al resto de mi familia.

Dedicatoria

A mis padres y hermano por su apoyo y comprensión.

A mi amiga Lisa Wright quien siempre ha estado aqui desde la distancia.

Resumen

En la Universidad de Sancti Spíritus “José Martí” (UNISS), en la actualidad, existe un grupo de investigación que profundiza en el campo de la inteligencia artificial y la computación afectiva. Varios son los resultados alcanzados hasta el momento pero sin tener en cuenta las emociones que presentan los estudiantes al trabajar con estas herramientas. Es por ello que el objetivo de este trabajo es desarrollar un Sistema Multiagente que logre revertir el miedo presente en los estudiantes durante su interacción con un ambiente de enseñanza-aprendizaje virtual. Para ello se seleccionaron las siguientes herramientas: PASSI como metodología del diseño, JADE como plataforma para la implementación de agentes, JAVA como lenguaje de programación, así como SQL para el desarrollo de la Base de Datos y SQLite para la gestión de la misma. Teniendo en cuenta lo anterior se puede concluir que el estudio de los fundamentos teórico metodológicos permitió contextualizar las diferentes variables que intervienen en el problema, lo cual posibilitó esclarecer que los agentes a utilizar en el Sistema Multiagente, no eran agentes inteligentes. Posibilitó apreciar el ilimitado campo de la programación afectiva, así como su importancia en el manejo de emociones, sobre todo en aplicaciones informáticas para la enseñanza. Se estudiaron varias metodologías pero PASSI se consideró la más adecuada pues abarca todo el ciclo de vida de un Sistema Multiagente, lo cual constituyó una guía efectiva para el desarrollo del mismo.

Abstract

In Sancti Spíritus University José Martí, nowadays, there is a researching group that deeps in the field of Artificial Intelligence and Affective Computing. Several results have been reached so far, but none taking into account the emotions the students might have while working with these tools. That´s why the object of this work is to develop a Multiagent System that achieve revert the fear that students might feel while working in a virtual teaching learning environment. To accomplish this were selected several tools such as: PASSI as design methodology, JADE as agent implementation platform, JAVA as programming language, adding SQL as Data Base developer and SQLite to manage it. Taking into account the above it can be conclude that theorical methodological foundations allowed contextualized the variables involved in the situation, which enabled clarify that the agents in the Multiagent System weren´t intelligents. Made possible to appreciate the unlimited field of affective programming, as well as its importance in the emotions management, must of all in teaching applications. Several methodologies were studied but PASSI was considered the most appropriated because its covers the whole life cycle of a Multiagent System, making it an effective guide through the process.

Tabla de contenido

Introducción:

Las emociones tienen su sede biológica en un conjunto de estructuras nerviosas denominado sistema límbico, que incluye el hipocampo, la circunvalación del cuerpo calloso, el tálamo anterior y la amígdala. La amígdala, además de desempeñar otras funciones, es la principal gestora de las emociones y su lesión anula la capacidad emocional. Las conexiones neuronales entre estas estructuras ubicadas en el cerebro reptiliano y la parte moderna del cerebro, el neocortex, son muchas y directas, lo cual asegura una comunicación de vértigo muy adaptativa en términos evolutivos (Ledoux, 1996)

De manera que se puede considerar que las emociones son tendencias de respuesta con un gran valor adaptativo, que tienen evidentes manifestaciones a nivel fisiológico, en la expresión facial, la experiencia subjetiva, el procesamiento de la información, etc., que son intensas pero breves en el tiempo y que surgen ante la evaluación de algún acontecimiento antecedente. (Jiménez, 2006)

En esta línea Fernández-Abascal y Palmero (1999) explican que el proceso emocional se desencadena por la percepción de condiciones internas y externas, que llegan a un primer filtro, la evaluación valorativa. Como consecuencia de esta valoración tiene lugar la actividad emocional que se compone de una experiencia subjetiva, una expresión corporal o comunicación no verbal, una tendencia a la acción o afrontamiento y cambios fisiológicos que dan soporte a todas las actividades anteriores. Las manifestaciones externas de la emoción o los efectos observables son frutos de un segundo filtro que tamiza las mismas y se refiere al aprendizaje y la cultura. Se valora si la situación es relevante, si el resultado es consistente o discordante con las expectativas, si es conducente u obstructivo para alcanzar las metas. Si la valoración que el sujeto realiza tiene valencia positiva, entonces, se puede afirmar que ese sujeto experimenta una emoción positiva. (Greco, Morelato, & Ison)

Algunos autores proponen como tipos de emociones positivas: la alegría y el amor. Fredrickson (1998) refiere cuatro tipos de emociones positivas: la alegría, el

interés, amor y satisfacción. Por su parte Pereyra señala la esperanza y Seligman (2003) describe el optimismo, Padrós Blázquez (2002) nombra la serenidad. Csiksentmiuhalyi (1998) describe el "flow" que en castellano podría traducirse como flujo de conciencia, definido como un estado en que la persona se encuentra completamente absorta en una actividad para su propio placer y disfrute durante el cual pierde la noción del tiempo y experimenta un enorme placer. (Greco et al.)

A pesar de la pujante aparición de la psicología positiva en los últimos años, el estudio de las "emociones negativas", miedo-ansiedad, ira y tristeza-depresión, tuvo durante todo el siglo XX, y sigue teniendo, mucha fuerza en la investigación psicológica. También añadimos el asco, que en los últimos 15-20 años ha sido objeto de interés por parte de la comunidad científica. Un concepto relacionado es el de afectividad negativa, que puede ser entendida como un estado emocional transitorio o como una diferencia persistente en el nivel general de afectividad. La afectividad negativa es un rasgo que refleja la tendencia a experimentar emociones negativas a través del tiempo y de situaciones (Watson & Clark, 1984). Este rasgo se solapa con el neuroticismo y la ansiedad, incluyendo sentimientos subjetivos de tensión, preocupación, ansiedad, ira y tristeza. (Rodríguez, 2009)

La expresión patológica del miedo son los trastornos por ansiedad, que están relacionados con una respuesta de ansiedad desproporcionada e irracional ante un peligro inexistente. Es una de las reacciones que produce mayor cantidad de trastornos mentales, conductuales, emocionales y psicosomáticos. La distinción entre fobia y miedo podría concretarse en que la reacción de miedo se produce ante un peligro real y la reacción es proporcionada a éste, mientras que en la fobia la respuesta de ansiedad es desproporcionadamente intensa (o innecesaria) con la supuesta peligrosidad del estímulo. (Rodríguez, 2009)

El miedo se define como una respuesta del organismo que se desencadena ante una situación de amenaza o peligro físico o psíquico, cuyo objeto es dotar al organismo de energía para anularlo o contrarrestarlo mediante una respuesta (conducta de huida o de agresión; Sandín & Chorot, 1995).

Psicológicamente, el miedo se combate mediante la exposición progresiva al objeto o sujeto en cuestión, de manera que la persona, progresivamente se dé cuenta que su miedo puede ser irracional y que no existe necesidad ninguna de tenerlo.

Esta investigación solo se referirá al miedo al fracaso, ese miedo que muchas veces está ligado al desconocimiento o simplemente a la falta de práctica, el miedo a no ser competentes cuando el resto de los compañeros parecen ser muy diestros y ágiles en el área o tarea.

Varias son las herramientas nacidas basadas en los principios de la computación afectiva cuyos objetivos finales han sido reconocer las emociones humanas. De los intentos que se han realizado para dotar a las computadoras de habilidad para expresar emociones se encuentran aquéllos sobre la representación de la dinámica de movimiento facial (Essa y Pentland, 1995), la síntesis de expresión oral afectiva (Cahn, 1990) y el uso de sistemas multiagente para inferir estados emocionales de tales agentes (Gmytrasiewicz y Lisetti, 2001).

Sin embargo ninguna de las anteriores ha ido más allá del mero hecho de identificar la emoción.

En Cuba la computación afectiva ha sido un campo poco explorado al igual que la utilización de los sistemas multiagente, aunque en este último sí se ha podido constatar resultados, pero en campos ajenos a la educación.

Es por ello que la Universidad de Sancti Spíritus "José Martí" un grupo de investigadores se proponen unir las ventajas de los SMA y de la Computación afectiva con el objetivo de mejorar el proceso de enseñanza-aprendizaje al interactuar con un ambiente virtual, pues hasta el momento se han desarrollado varios software para apoyar este proceso pero sin tener en cuenta las emociones que presenta el estudiante, las cuales pueden ser muy dañinas para el proceso de asimilación del conocimiento del educando.

Partiendo de la situación anterior se puede definir como **problema de esta investigación** al siguiente:

¿Cómo revertir el Miedo como emoción negativa presente en los estudiantes universitarios durante su interacción con un ambiente de enseñanza aprendizaje virtual?

Para dar solución a este problema se ha trazado como **objetivo general**: Desarrollar un Sistema Multiagente (SMA) para revertir la Miedo como emoción negativa presente en los estudiantes universitarios durante su interacción con un ambiente de enseñanza aprendizaje virtual.

Para cumplir con este objetivo se definieron las siguientes preguntas de investigación:

1. ¿Cuáles son los fundamentos teóricos, metodológicos que permiten desarrollar un Sistema Multiagente para revertir la emoción negativa Miedo presente en los estudiantesuniversitarios durante su interacción con un ambiente de enseñanza aprendizaje virtual?
2. ¿Cómo diseñar un Sistema Multiagente para revertir la emoción negativa Miedo presente en los estudiantes universitarios durante su interacción con un ambiente de enseñanza aprendizaje virtual?
3. ¿Cómo implementar un Sistema Multiagente para revertir la emoción negativa Miedo presente en los estudiantes universitarios durante su interacción con un ambiente de enseñanza aprendizaje virtual?

Para dar solución a estas preguntas de investigación se trazaron las siguientes tareas de investigación:

1. Determinación de los fundamentos teóricos, metodológicos que permitan implementar un Sistema Multiagente para revertir la emoción negativa Miedo en los estudiantes universitarios durante su interacción con un ambiente de enseñanza aprendizaje virtual.

2. Diseño de un Sistema Multiagente para revertir la emoción negativa Miedo presente en los estudiantes universitarios durante su interacción con un ambiente de enseñanza aprendizaje virtual.
3. Implementación de un Sistema Multiagente para revertir la emoción negativa Miedo presente en los estudiantes universitarios durante su interacción con un ambiente de enseñanza aprendizaje virtual.

Este trabajo estará compuesto por una introducción, tres capítulos, conclusiones y recomendaciones y en él la investigación profundizará en el tema de la siguiente forma:

- En el Capítulo 1. Se exponen las técnicas de la Inteligencia Artificial para el desarrollo de Sistemas Multiagente en función de la Programación Afectiva, se presentan los avances más significativos que recoge la literatura sobre la obtención de modelos, herramientas, metodologías y plataformas para la implementación de Sistemas Multiagente.
- En el Capítulo 2. Diseño de la solución propuesta con PASSI (Process for Agent Societies Specification and Implementation), se ofrece la aplicación de la metodología seleccionada a través de todas sus etapas y modelos relacionados para obtener la solución que se propone.
- En el Capítulo 3. Implementación de la aplicación con JADE (Java Agent DEvelopment Framework) y Netbeans.

Capítulo 1. Herramientas para el desarrollo de un SMA

En este capítulo se contextualizan las variables asociadas a la computación afectiva, los agentes, los SMA y se profundiza en el miedo como emcoción negativa y sus tratamientos. También se referirá a las diferentes plataformas de diseño e implementación que existen para el desarrollo de esta aplicación de software y los lenguajes utilizados para ello.

1.1 Computación Afectiva

La Computación Afectiva es una disciplina de la Inteligencia Artificial que intenta desarrollar métodos computacionales orientados a reconocer emociones humanas y generar emociones sintéticas, es un campo emergente cuyo objetivo es el desarrollo de sistemas inteligentes capaces de dotar a un ordenador con la habilidad de reconocer, interpretar, procesar y expresar emociones.

La fundadora de esta línea de trabajo es Rosalind Picard, investigadora del M.I.T. (Massachusetts Institute of Technology), quién publico el libro “Affective Computing” en el año 2000.

Esta disciplina surge frente a la necesidad de optimizar la interacción entre personas y computadoras, pero también se inscribe en la investigación de los procesos inteligentes. Como aclara dicha autora, las emociones son una parte muy importante de nuestras decisiones (aún de las que parecen más “racionales”). Como prueba de esto, Picard expone casos en donde, personas que sufrieron lesiones en regiones del cerebro asociadas a las emociones, sufrían consecuentemente cierta incapacidad frente a la toma de decisiones y determinados razonamientos lógicos. Teniendo en cuenta, entonces, el lugar que parece ocupar las emociones en los procesos inteligentes, R. Picard propone que, a la hora de modelar procesos inteligentes, deberemos tener en cuenta a los procesos emocionales y la forma en que éstos participan en la inteligencia.

Se pueden plantear dos problemáticas de las que se ocupa la Computación Afectiva:

- El reconocimiento de emociones (y de expresiones emotivas) humanas por parte de una computadora.
- La simulación (o generación) de estados y expresiones emocionales con computadoras.

Por poner un ejemplo, una persona que haya sufrido un desengaño sentimental puede mostrar un comportamiento superficial normal en, digamos, las redes sociales. Sin embargo, su ratio de publicación de mensajes, gestos e intensidad pueden servir de inspiración para que, por ejemplo, se le sugieran páginas de contactos y relaciones, ofertas de helado de chocolate o libros de autoayuda para afrontar una ruptura.

La utilidad principal de un Computador Afectivo es la de interactuar con el usuario de una manera lo más eficiente y cómoda posible. Según Piccard, los computadores no necesitan habilidades afectivas para convertirse en una especie de humanoides, si no que las necesitan para funcionar con más inteligencia, sensibilidad y efectividad respecto a los seres humanos. (Peñaranda, 2015)

La computación afectiva ha demostrado su utilidad en el tratamiento de trastornos como el autismo, el síndrome de Asperger, la epilepsia y depresión, así como en el reconocimiento de riesgo de estrés y su mitigación. También tiene aplicación en medición y monitorización de la temperatura corporal, indicadores fisiológicos, sueño, ritmo cardiaco… en adultos, bebés, pacientes, drogodependientes, etc. (Peñaranda, 2015)

1.2 Influencia de la computación afectiva en la educación

Lo importante no es que una máquina tenga o no emociones: no se buscan sistemas que estén tristes o alegres, sino que puedan enfocar nuestra atención y mejorar la toma de decisiones, adaptándose al contexto del momento. Por

ejemplo, un estudiante motivado y siempre acompañado de un buen estado de humor, también será un estudiante con más posibilidades de aprendizaje y retención de datos.

Temas puramente cognitivos, como la resolución de problemas, son importantes, pero los aspectos “afectivos” del aprendizaje se consideran más cuidadosamente debido a que se reconoce que los estados emocionales de los estudiantes tienen una gran influencia sobre las posibles metas de logro establecidas a priori y durante el proceso de aprendizaje. No obstante, no existe una respuesta a qué tipo de modelado afectivo se necesita incluir en los sistemas para mejorar el aprendizaje del estudiante.

Rosalind Picard (1997) clasifica los tipos de características que serían necesarias para proveer a las computadoras con los medios para tomar en cuenta los estados afectivos de los estudiantes como sigue:

- Reconocer emociones: Este aspecto es considerado como la habilidad para inferir el estado emocional a partir de observaciones de las expresiones y por medio del razonamiento acerca de una situación generadora de emociones.
- Expresar emociones: El requerimiento básico para que una computadora pueda expresar emociones es que tenga canales de comunicación como voz o imagen y la habilidad para comunicar información afectiva sobre esos canales.
- Tener emociones: Para cumplir con esta característica, es necesario cubrir aspectos como tener emociones primarias (aquellas ligadas a respuestas innatas a eventos potencialmente dañinos como el enojo y el miedo); tener emociones generadas cognitivamente (que involucran el razonamiento cognitivo explícito en su generación).
- Tener inteligencia emocional: Esta característica podría observarse en una computadora que no sólo supiera expresar y tener emociones, sino que también supiera cómo administrar su expresión y cómo utilizar sus emociones para pensamiento creativo y motivación.

De los intentos que se han realizado para dotar a las computadoras de habilidad para expresar emociones se encuentran aquéllos sobre la representación de la dinámica de movimiento facial (Essa y Pentland, 1995), la síntesis de expresión oral afectiva (Cahn, 1990) y el uso de sistemas multiagente para inferir estados emocionales de tales agentes (Gmytrasiewicz y Lisetti, 2001).

Así, entre los campos relacionados con la investigación de aspectos afectivos en las computadoras se encuentran: a) Identificar los estados emotivos del alumno relevantes al aprendizaje; b) cuáles son los dominios de aprendizaje en los que dichos estados afectivos son relevantes; c) qué acciones pedagógicas son las pertinentes considerando dichos estados afectivos; d) cuál es la inteligencia social y emocional adecuada para dichos estados afectivos del estudiante por parte del tutor; e) de qué manera hay que considerar la inteligencia emocional social y colectiva.

Una de las metas para generar aplicaciones por computadora que consideren los estados afectivos del estudiante consiste en propiciar las condiciones adecuadas para que pueda alcanzarse el aprendizaje. Sin embargo, esta tarea no es nada fácil y puede tratarse desde distintas perspectivas (como las que identifica Piccard: reconocer, expresar, tener y gestionar las emociones). Modelar ciertos aspectos afectivos del estudiante puede coadyuvar a brindarle una mejor situación de aprendizaje y, por tanto, es loable continuar con el esfuerzo.

Para lograr aplicaciones por computadora que consideren los estados afectivos de los estudiantes e influyan en su rendimiento académico, en la actualidad muchos desarrolladores se apoyan en el uso de Agentes Inteligentes o Sistemas Multiagente.

1.3 Agente

Se puede definir al agente inteligente como una entidad de software que, basándose en su propio conocimiento, realiza un conjunto de operaciones destinadas a satisfacer las necesidades de un usuario o de otro programa, bien

por iniciativa propia o porque alguno de éstos se lo requiere. (Hípola & Vargas-Quesada, 1999)

Todos los agentes inteligentes son programas, pero no todos los programas que realizan búsquedas son agentes inteligentes. Los agentes en sí mismos pueden ser considerados como entidades individuales (partes de programa que tienen control sobre sus propias vidas y movimientos). Continuamente están realizando procesos que les indican qué hacer y cómo. Se comunican con otros agentes para resolver de forma adecuada su trabajo. (Hípola & Vargas-Quesada, 1999)

De acuerdo con el punto de vista de la inteligencia artificial un agente posee las siguientes propiedades que se describen a continuación:

- Autonomía: actuar sin ningún tipo de intervención humana directa, y tener el control sobre sus propios actos.
- Situación: Situarse dentro de un entorno, ya sea real o virtual.
- Sociabilidad: comunicarse por medio de un lenguaje común con otros agentes, e incluso con los humanos.
- Inteligencia: Rodearse de conocimiento (creencias, deseos, intenciones y metas).
- Capacidad de reacción: percibir su entorno, y reaccionar para adaptarse a él.
- Organización: Organizarse dentro de sociedades que siguen unas estructuras similares a las definidas en sociedades humanas o ecológicas.
- Iniciativa: emprender las acciones para resolver un problema.

Pero, como todo sistema creado por el hombre, un agente inteligente posee ventajas y desventajas. Dentro de las ventajas podemos encontrar que, tienden a facilitarles el trabajo a los usuarios, actúan como consultantes, y, sirven de operadores en medios complejos. Mientras que en sus desventajas podemos encontrar que, a veces es difícil diferenciar entre la información relevante y la irrelevante, cómo agilizar la búsqueda, cómo evitar repetir una tarea realizada.

Pero, cuando se tiene un conjunto de agentes autónomos, generalmente heterogéneos y potencialmente independientes, que trabajan en común resolviendo un problema, capaces de tomar la iniciativa, capaces de compartir conocimiento, capaces de cooperar y negociar, capaces de comprometerse con metas comunes; estamos en presencia de Sistemas Multiagente (MAS).

1.4 Sistema Multiagente

Un sistema multiagente (SMA) es un sistema compuesto por múltiples agentes inteligentes que interactúan entre ellos. Los sistemas multiagentes pueden ser utilizados para resolver problemas que son difíciles o imposibles de resolver para un agente individual o un sistema monolítico.

Los agentes en un sistema multiagente tienen varias características importantes:

- Autonomía: los agentes son al menos parcialmente autónomos
- Visión local: ningún agente tiene una visión global del sistema, o el sistema es demasiado complejo para un agente para hacer un uso práctico de esos conocimientos
- Descentralización: no hay un agente de control designado (o el sistema se reduciría a un sistema monolítico)

Normalmente la investigación de sistemas multiagente se refiere a agentes de software. Sin embargo, los agentes en un sistema multiagente también podrían ser robots, seres humanos o equipos humanos.

Los SMA se desarrollan sobre middleware y proporcionan un nuevo nivel de abstracción más intuitivo. El diseño de SMA, generalmente, se aborda pensando en los agentes como entes con motivación. En lugar de modelar un sistema con componentes que ejecutan métodos, el desarrollador tiene que pensar en los objetivos que los componentes deben alcanzar y en las tareas necesarias para que lo consigan. Al desarrollar los componentes así, se espera que el proceso sea más intuitivo ya que esta forma de modelar y de razonar se halla más cerca del

pensamiento humano que los paradigmas de programación tradicionales. (Sanz, 2003)

La construcción de SMA integra tecnologías de distintas áreas de conocimiento: técnicas de ingeniería del software para estructurar el proceso de desarrollo; técnicas de inteligencia artificial para dotar a los programas de capacidad para tratar situaciones imprevistas y tomar decisiones, y programación concurrente y distribuida para tratar la coordinación de tareas ejecutadas en diferentes máquinas bajo diferentes políticas de planificación. (Sanz, 2003)

1.5 Metodologías para el diseño de SMA

Con el auge de la Inteligencia Artificial Distribuida, el desarrollo de agentes y SMA, se han creado infinidad de metodologías que hacen posible el diseño de un SMA, facilitando así su futura implementación. Algunas de estas metodologías son:

GAIA es una metodología para el diseño de sistemas basados en agentes cuyo objetivo es obtener un sistema que maximice alguna medida de calidad global. GAIA pretende ayudar al analista a ir sistemáticamente desde unos requisitos iniciales a un diseño que, según los autores, esté lo suficientemente detallado como para ser implementado directamente.

GAIA propone trabajar inicialmente con un análisis a alto nivel. En este análisis se usan dos modelos, el modelo de roles para identificar los roles clave en el sistema junto con sus propiedades definitorias y el modelo de interacciones que define las interacciones mediante una referencia a un modelo institucionalizado de intercambio de mensajes, como el FIPA-Request. Tras esta etapa, se entraría en lo que GAIA considera diseño a alto nivel. El objetivo de este diseño es generar tres modelos: el modelo de agentes que define los tipos de agente que existen, cuántas instancias de cada tipo y qué papeles juega cada agente, el modelo de servicios que identifica los servicios (funciones del agente) asociados a cada rol, y un Modelo de conocidos, que define los enlaces de comunicaciones que existen entre los agentes.

MaSE (Multi-agent Systems Software Engineering) parte del paradigma orientado a objetos y asume que un agente es sólo una especialización de un objeto. La especialización consiste en que los agentes se coordinan unos con otros vía conversaciones y actúan proactivamente para alcanzar metas individuales y del sistema.

El análisis en MaSE consta de tres pasos: capturar los objetivos, capturar los casos de uso y refinar roles. Como productos de estas etapas se esperan: diagramas de objetivos, que representan los requisitos funcionales del sistema; diagramas de roles, que identifica roles, tareas asociadas a roles y comunicaciones entre roles y entre tareas; y casos de uso, mostrados no como diagrama sino como una enumeración de los casos de uso considerados con la posibilidad de usar diagramas de secuencia para detallarlos.

El diseño consta de cuatro pasos: crear clases de agentes, construir conversaciones, ensamblar clases de agentes y diseño del sistema. Como productos de estas etapas, MaSE espera: diagramas de clases de agentes, que enumeran los agentes del sistema, roles jugados e identifican conversaciones entre los mismos; descomposición del sistema (agente) en subsistemas (componentes del agente) e interconexión de los mismos (definición de la arquitectura del agente mediante componentes); diagramas UML de despliegue para indicar cuántos agentes habrá en el sistema y de qué tipo.

PASSI

Dentro de esta metodología podemos identificar varias fases con las cuales se asegura llegar de la especificación del sistema a una versión implementable. Dentro de la etapa de análisis, tenemos las siguientes fases: descripción del dominio: utilizando los diagramas de casos de uso de UML se describe el dominio de la aplicación, identificación de agentes: a partir del diagrama anterior y utilizando Rational Rose identificamos los agentes que componen el sistema, identificación de roles: los roles de los agentes los representamos mediante diagramas de secuencia e identificación de tareas: se dibuja un diagrama de

actividad para cada agente y se deciden que tareas son necesarias para realizar las funcionalidades descritas anteriormente.

En la segunda etapa, la etapa de diseño, podemos identificar otras fases entras cuales se encuentran: descripción de ontologías: se describe la sociedad desde el punto de vista ontológico, descripción de roles: se modela la vida de los agentes observando sus roles, descripción de protocolos: se define cuales protocolos se utilizarán y si es necesario, se definirán los nuevos, definición de la estructura y del comportamiento de los agentes: se diseña la estructura de la sociedad y de los agentes en particular.

Finalmente tenemos las fases de implementación y configuración. La fase de producción de código no está completamente implementada, pero está en desarrollo la búsqueda de una solución en un metalenguaje basado en XML. En cuanto a la configuración, se acentúa su importancia si se trabaja con agentes móviles y con problemas significativos en la diseminación de los agentes en el sistema.

(Marchetti & García)

1.6 Plataformas para el desarrollo de agentes

En la actualidad, existen infinidad de plataformas para el desarrollo de agentes y específicamente que se dedican a la rama de la computación afectiva. A continuación se muestran las características principales de algunos de ellos considerados los más importantes:

JACK

JACK provee un entorno de desarrollo orientado a agentes construido sobre Java y completamente integrado con este lenguaje de programación. Incluye todos los componentes del entorno de desarrollo de Java así como también ofrece extensiones específicas para implementar el comportamiento de los agentes. La relación entre JACK y Java es análoga a la relación entre los lenguajes C++ y C, JACK fue desarrollado para proveer extensiones orientadas a agentes al lenguaje

Java. El código JACK es primero compilado a código Java regular antes de ser ejecutado. El lenguaje JACK Agent hace más que extender la funcionalidad de Java - provee un entorno para soportar un nuevo paradigma de programación. Este lenguaje es orientado a los agentes y es utilizado para implementar sistemas de software orientados a agentes. Todas las formas en las que extiende a Java, son implementadas como plug-ins, lo que permite que el lenguaje sea lo más extensible y flexible posible. Les extensiones son las siguientes:

define nuevas clases base, interfaces y métodos

provee extensiones a la sintaxis de Java para soportar clases orientadas a agentes

provee extensiones semánticas para soportar la ejecución del modelo

JADE (Java Agent Development Framework)

JADE es un entorno que simplifica la implementación de sistemas multiagente mediante una capa de soporte (middle-ware) que respeta las especificaciones FIPA y con un conjunto de herramientas para el desarrollo y debugging. La plataforma puede ser distribuida en varias máquinas (las cuales no necesitan compartir el mismo sistema operativo) y la configuración puede ser controlada mediante una interface gráfica remota. La configuración puede incluso ser cambiada en tiempo de ejecución moviendo agentes de una máquina a otra, cuando es necesario.

La arquitectura de comunicación ofrece mensajes flexibles y eficientes, mientras que JADE crea y maneja una cola de mensajes ACL entrantes; los agentes pueden acceder su cola mediante una combinación de varios modos: blocking, polling, timeout y pattern matching. El modelo de comunicación de FIPA ha sido implementado completo y sus componentes han sido claramente distinguidas y completamente integradas: protocolos de interacción, ACL, lenguaje de contenido, esquemas de codificación, ontologías y finalmente protocolos de transporte. El mecanismo de transporte, en particular, es como un camaleón debido a que se

adapta a cada situación, seleccionando transparentemente el mejor protocolo disponible entre RMI de Java, notificación de eventos e IIOP.

JAFMAS (Java Framework for Multi-agent Systems)

JAFMAS provee una metodología genérica para desarrollar sistemas multiagente basados en los actos del habla junto con un conjunto de clases para soportar la implementación de estos agentes en Java. La intención del framework es asistir a los desarrolladores principiantes y expertos a estructurar sus ideas en aplicaciones de agentes concretas. La metodología está basada en cinco etapas:

1. Identificación de agentes
2. Definición de las conversaciones de cada agente
3. Determinación de las reglas que gobiernan las conversaciones de cada agente
4. Analizar la coherencia entre todas las conversaciones en el sistema
5. Implementación

El soporte para la comunicación está provisto para ambos casos de comunicación, directo y broadcast basado en sujetos. El soporte lingüista es provisto por los lenguajes de comunicación basados en los actos del habla. El soporte de coordinación proviene de la conceptualización de los planes de un agente y su coordinación como conversaciones basadas en reglas representadas mediante modelos de autómatas.

MADKit (Multi-agent Development Kit)

MADKit es una plataforma multiagente para desarrollar y ejecutar aplicaciones basadas en un paradigma orientado a la organización. Estos paradigmas multiagente utilizan agentes, grupos y roles como los puntos bases para construir aplicaciones complejas. MADKit no fuerza ninguna consideración acerca de la

estructura interna de los agentes, de esta manera permite a los desarrolladores implementar libremente sus propias arquitecturas de agentes.

MADKit está construido alrededor del concepto de micro-kernel y agentificación de servicios. El kernel de MADKit es más bien pequeño, pero los agentes ofrecen los servicios importantes que se pueden necesitar para las aplicaciones. Distribución y pasaje remoto de mensajes, monitoreo y observación de agentes, edición, etc..., son todas realizadas por agentes. Es más, MADKit provee un conjunto de "contenedores", esto es entornos de ejecución para correr aplicaciones, con el objetivo de que MADKit trabaje en diferentes situaciones: como un entorno de desarrollo, pero también como una herramienta embebida para aplicaciones.

ZEUS

El objetivo del proyecto ZEUS es facilitar el desarrollo rápido de nuevas aplicaciones multiagente mediante la abstracción de los principios y componentes más comunes a una herramienta. La idea es proveer una herramienta de propósito general y personalizable, que permita la creación de agentes colaborativos y que pueda ser usada por ingenieros de software con poca experiencia en tecnología de agentes para crear sistemas multiagente. Para esto es necesario cumplir con los siguientes principios:

- La herramienta debe permitir separar el problema de nivel de dominio y la funcionalidad de nivel de agente
- La herramienta debe estar basada en el paradigma de programación visual
- La herramienta debe soportar un diseño abierto para asegurar que sea fácilmente extensible
- Se debe utilizar tecnología "estandarizada" siempre que sea posible

Los agentes deben ser deliberativos en el sentido de que deben explícitamente razonar acerca de sus acciones en términos de que metas seguir, cuando considerar nuevas metas y cuando abandonar metas existentes. Es más, el requerimiento del comportamiento dirigido por metas implica que el agente solo seleccionará acciones que espera de alguna manera lo acerque a la meta

deseada. Sólo se descartan metas cuando, no son alcanzables o las motivaciones para alcanzar dicha meta ya no se verifican.

(Marchetti & García)

1.7 Programación orientada a agentes

La construcción de un SMA implica una visión diferente del proceso de programación: debe incluir y aprovechar las características propias de los agentes. El uso de agentes para la solución de un problema lleva en forma natural al concepto de Programación Orientada a Agentes (POA), la cual aparece como un nuevo paradigma de programación. Una de las características deseables implícitas en la aplicación del paradigma POA es que facilita el desarrollo de aplicaciones complejas. En efecto, subdividir un problema complejo para que sea resuelto por entidades activas, los agentes, brinda la posibilidad de diseñar una solución modular que permite un manejo más estructurado y coherente de la complejidad global. Sin embargo, es claro, que no al dividir el sistema en partes surgen múltiples problemas para lograr que los agentes trabajen en forma cooperativa en pro de los objetivos del sistema. (Ahogado, Reinemer, & González, 2003)

En cierta forma se puede afirmar que la POA es una evolución de la programación orientada a objetos (POO). Los agentes tienen características similares a los objetos, ambos son entidades que encapsulan datos y comportamiento; los dos permiten generar programas modulares que incluyan mecanismos de herencia con alto potencial de reutilización. La diferencia más importante radica en la proactividad y la autonomía de los agentes. Un agente es un ente activo que actúa autónomamente, sin necesidad de ser invocado, para cumplir una tarea. Desde el punto de vista de implementación, para el caso de agentes no físicos como los robots, un agente es un proceso/hilo que realiza una tarea; y por tanto la programación de un SMA incluye elementos propios de la programación concurrente. Un agente retiene las características de encapsulamiento y modularidad de los objetos, y al mismo tiempo aprovecha las ventajas del

paralelismo propio de los sistemas concurrentes. Sin embargo, el punto fuerte de los sistemas de agentes, es que guardan el sentido semántico propio de su definición conceptual originaria de la inteligencia artificial; la semántica es más elaborada y de mayor nivel de abstracción, lo cual permite abordar la problemática de construcción de un sistema con elementos nuevos y desde una perspectiva más amplia. (Ahogado et al., 2003)

JavaLog es un lenguaje de programación que combina los paradigmas de orientación a objetos y lógicos a través de la utilización de Java y Prolog. En el proceso de integrar estos lenguajes para facilitar la programación de agentes se ha desarrollado un intérprete Prolog en el lenguaje Java con el fin de posibilitar extensiones del mismo a través de sub-clasificación. (Zunino, Berdún, & Amandi, 2001)

Programar agentes con JavaLog es programar un agente como un objeto Java, el cual es instancia de una clase que representa ese tipo de agente. La funcionalidad del agente es implementada en métodos codificados básicamente en Java. (Zunino et al., 2001)

1.8 Lenguajes de programación

Los lenguajes de programación son idiomas artificiales diseñados para expresar cálculos y procesos que serán llevados a cabo por ordenadores. Un lenguaje de programación está formado por un conjunto de palabras reservadas, símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. El proceso de programación consiste en la escritura, compilación y verificación del código fuente de un programa.(Guevara)

Para la implementación de esta herramienta se utilizará el lenguaje de programación Java, el cual es un lenguaje de programación desarrollado por Sun Microsystems. Java fue presentado en la segunda mitad del año 1995 y desde entonces se ha convertido en un lenguaje de programación muy popular. Java es un lenguaje muy valorado porque los programas Java se pueden ejecutar en diversas plataformas con sistemas operativos como Windows, Mac OS, Linux o

Solaris. James Gosling, el director del equipo de trabajo encargado de desarrollar Java, hizo realidad la promesa de un lenguaje independiente de la plataforma. Se buscaba diseñar un lenguaje que permitiera programar una aplicación una sola vez que luego pudiera ejecutarse en distintas máquinas y sistemas operativos. Para conseguir la portabilidad de los programas Java se utiliza un entorno de ejecución para los programas compilados. Este entorno Java Runtime Environment (JRE). Es gratuito y está disponible para los principales sistemas operativos. Esto asegura que el mismo programa Java pueda ejecutarse en Windows, Mac OS, Linux o Solaris.

Los programas Java se compilan a un lenguaje intermedio, denominado Bytecode. Este código es interpretado por la máquina virtual de Java del entorno de ejecución (JRE) y así se consigue la portabilidad en distintas plataformas. (Guevara)

Otro de los lenguajes a utilizar para la implementación de esta herramienta es SQL (Structure Query Language), el cual es un lenguaje de consulta estructurado establecido claramente como el lenguaje de alto nivel estándar para sistemas de bases de datos relacionales. Los responsables de publicar este lenguaje como estándar, fueron precisamente los encargados de publicar estándar, la ANSI (Instituto Americano de Normalización) y la ISO (Organismo Internacional de Normalización). (García, 2003)

El SQL es un lenguaje muy parecido al lenguaje natural; concretamente, se parece al inglés, y es muy expresivo. Por estas razones, y como lenguaje estándar, el SQL es un lenguaje con el que se puede acceder a todos los sistemas relacionales comerciales. (Escofet)

Como gestor para esta base de datos se escogió SQLite, la cual es una base de datos muy similar a la conocida Access del mundo Windows pero a diferencia de esta posee una serie de ventajas que la hacen interesante de aplicar. Para comenzar es multiplataforma y cumple con los estándares (en su mayoría) SQL92 por lo que su sintaxis y forma de uso casi no posee curva de aprendizaje a los

conocedores de SQL y sus MySql, porque además como este último (implementación de mysql en php) sqlite también está contemplada en el tratamiento dinámico de php profusamente. (Aguiar, 2006)

1.9 Cómo tratar el miedo desde la programación afectiva

La mejor manera de combatir esta emoción, puede ser, proveer a los estudiantes de una herramienta (video tutorial), que les facilite el uso de la misma, que los haga sentir seguros y borre ese rastro de vergüenza al pensar que son menos capacitados por no haber aprendido a usar ese software, programa o aplicación antes.

El uso de mensajes alentadores puede ser una manera de estímulo también aceptada de manera positiva por los estudiantes, los cuales siempre estarán fascinados por el modo de respuesta de la computadora, y de la cual, llegarán a sentir más afecto que cualquier otro compañero, pues será menos vergonzoso recurrir a ella por ayuda.

1.10 Conclusiones:

En este capítulo se definieron conceptos básicos ligados a la inteligencia artificial, la programación afectiva y la programación orientada a agentes. Además, se describieron las herramientas a utilizar en el diseño e implementación del Sistema Multiagente.

Capítulo 2. Diseño del SMA utilizando PASSI

En este capítulo se analizarán los dos primero niveles de PASSI con el objetivo de diseñar el SMA, definiendo para ello los diferentes modelos que sugiere la metodología utilizando el PASSI TOOL KIT.

2.1 Modelo de Requerimientos del Sistema

2.1.1 Descripción del Entorno

Este modelo proporciona una visión al máximo nivel de abstracción del sistema, en él se definen dos tipos de actores: actores internos y actores externos. Como actor interno tenemos al estudiante que interactúa de manera directa con el sistema, mientras que como actor externo está la PC, la cual tendrá todos los recursos externos necesarios para revertir las emociones negativas, también identificado como CU.

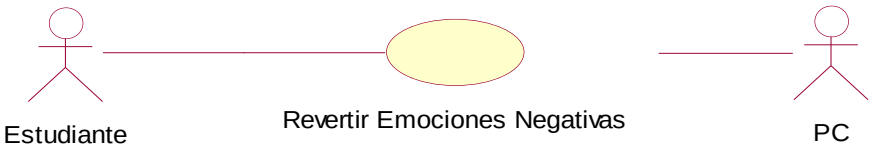

Ilustración 1: Diagrama de Entorno

2.1.2 Descripción del Dominio

Este diagrama ofrece una descripción funcional del sistema usando Diagramas de CU. Describe los requerimientos del sistema en una serie jerárquica de Diagramas de CU. En él, debe representarse al menos un CU por cada agente que se pretenda identificar. Constituye un primer acercamiento a lo que será el sistema en sí.

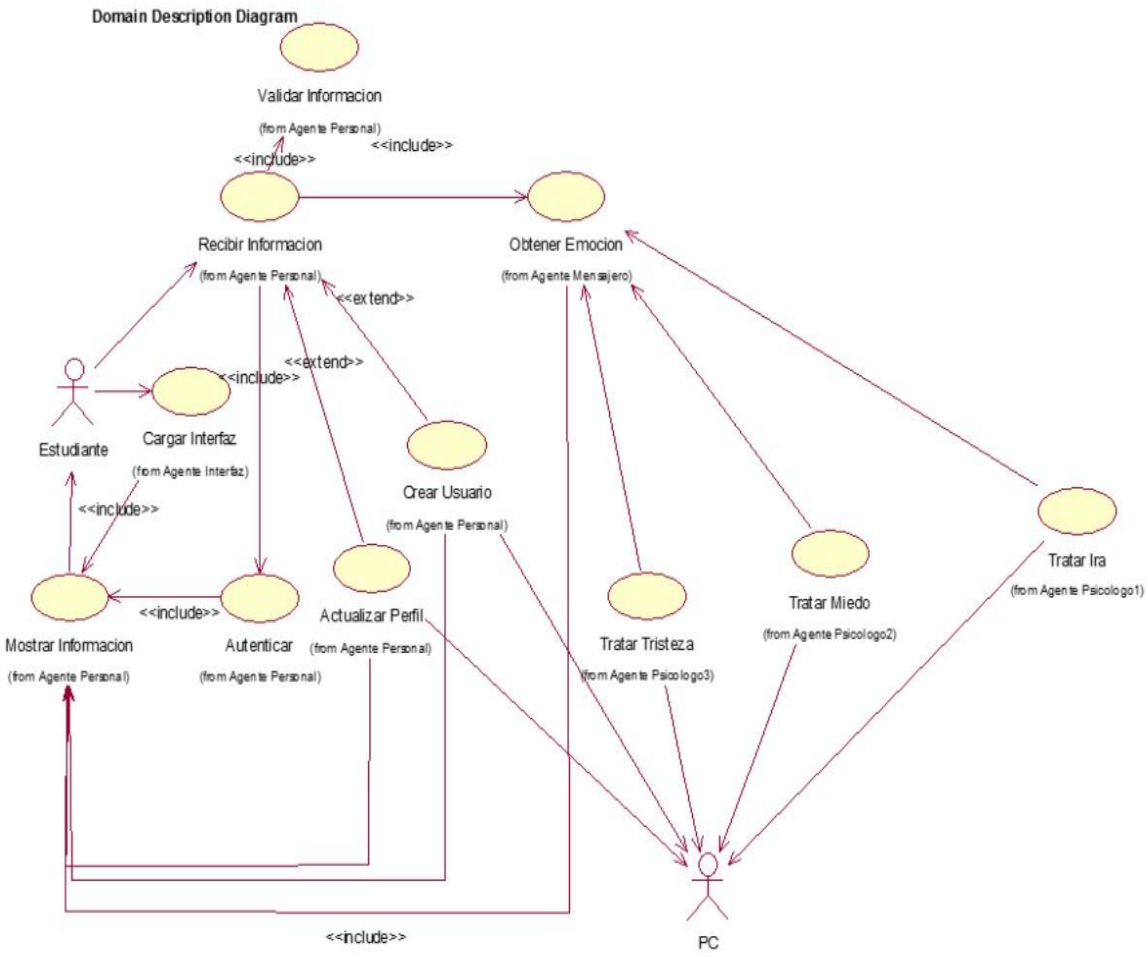

Ilustración 2: Diagrama de Descripción del Dominio

Los Casos de Uso identificados fueron:

- Cargar Interfaz
- Mostrar Información
- Autenticar
- Recibir Información
- Validar Información
- Crear Usuario

- Actualizar Perfil
- Obtener Emoción
- Tratar Ira
- Tratar Miedo
- Tratar Tristeza

Tabla 1: Especificación de CU: Autentificar

Caso Uso: Autentificar		
Descripción:	Este CU tiene como objetivo que el estudiante se autentifique al iniciar su sesión con el sistema, se verificarán sus datos y se les permitirá el acceso en caso de que los datos estén bien, en caso contrario el sistema le solicitará que vuelva a autentificarse.	
Comunicaciones:	**Iniciador**	**Participantes**
	1. CU Autentificar.	1. CU Validar Información. 2. CU Recibir Información.

Tabla 2: Especificación de CU: Crear Usuario

Caso Uso: Crear Usuario		
Descripción:	Este CU tiene como objetivo que el estudiante se registre en el software si nunca ha interactuado con él, y posteriormente gestione sus preferencias.	
Comunicaciones:	**Iniciador**	**Participantes**
	1. CU Crear Usuario.	1. CU Validar Información. 2. CU Recibir Información.

Tabla 3: Especificación de CU: Validar Información

Caso Uso: Validar Información		
Descripción:	Al introducir los datos en el sistema este es el encargado de verificar que la información introducida sea correcta.	
Comunicaciones:	**Iniciador**	**Participantes**
	1. CU Autentificar.	1. CU Validar

		Información.

Tabla 4: Especificación de CU: Actualizar Perfil

Caso Uso: Actualizar Perfil		
Descripción:	Este CU tiene como objetivo permitir al estudiante cambiar algunos de sus datos como usuario, siendo el usuario el único dato imposible de modificar.	
Comunicaciones:	**Iniciador**	**Participantes**
	1. CU Actualizar Perfil	1. CU Recibir información 2. CU Validar Información.

Tabla 5: Especificación de CU: Obtener Emoción

Caso Uso: Obtener Emoción		
Descripción:	Este CU tiene como objetivo obtener la emoción seleccionad por el estudiante para poder comenzar su tratamiento.	
Comunicaciones:	**Iniciador**	**Participantes**
	1. CU Autentificar 2. CU Obtener emoción	1. CU Tratar Ira 2. CU Tratar Miedo. 3. CU Tratar Tristeza.

Tabla 6: Especificación de CU: Tratar Miedo

Caso Uso: Tratar Miedo		
Descripción:	Este CU tiene como objetivo tratar el miedo que experimente el estudiante, este CU estará compuesto por 2 tratamientos.	
Comunicaciones:	**Iniciador**	**Participantes**
	1. CU Obtener Emoción.	1. CU Tratar Miedo.

2.1.3 Identificación de Agentes

Este diagrama del diseño permite la separación de responsabilidades en agentes.

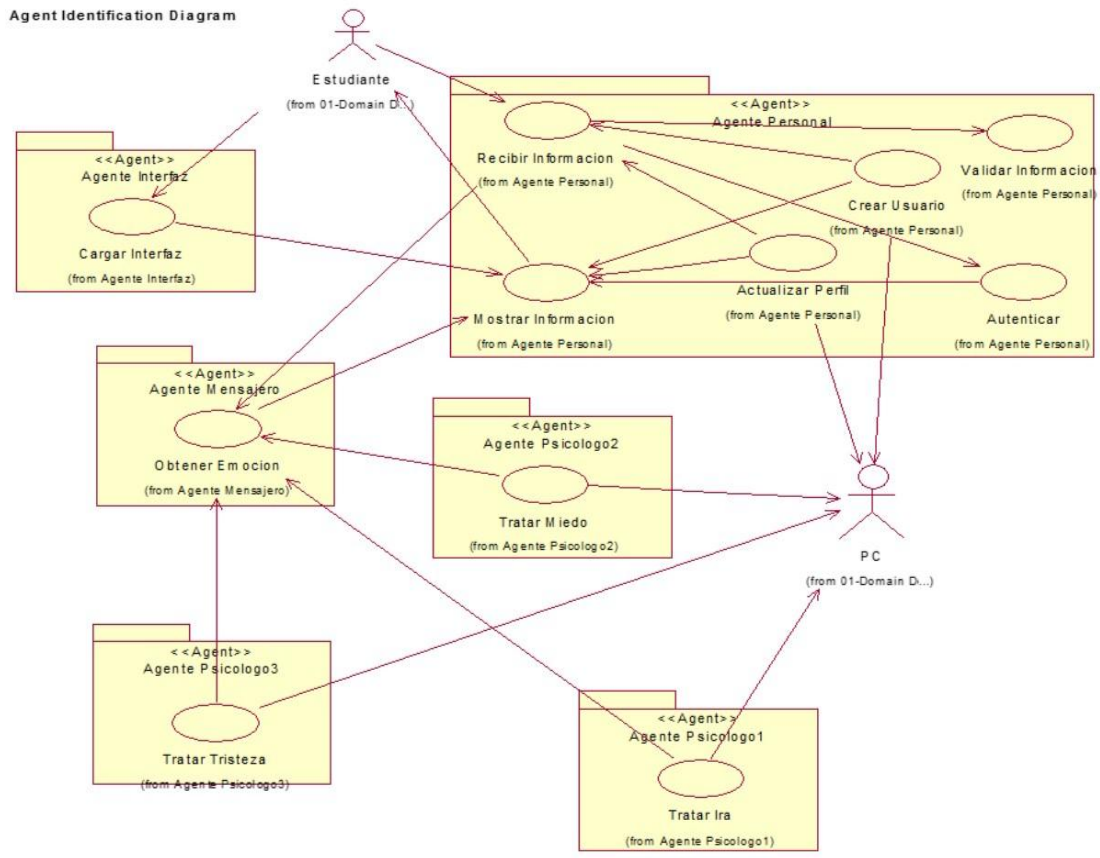

Ilustración 3: Diagrama de Identificación de Agentes

En este diagrama fueron identificados 6 agentes como parte del proceso de tratamiento de emociones negativas:

Agente Interfaz

1. CU Cargar Interfaz

Este CU será el encargado de una vez el usuario inicie el software, llamar al agente personal, levantando así la interfaz principal.

Agente Personal

1. CU Mostrar Información

2. CU Autenticar
3. CU Recibir Información
4. CU Validar Información
5. CU Crear usuario.
6. CU Actualizar perfil

Este agente será el encargado del manejo de todos los datos concernientes al usuario y de mostrar y recibir esos datos y las confirmaciones.

Agente Mensajero

1. CU Obtener emoción

Este será el encargado de recibir la emoción seleccionada por el usuario y de acuerdo a esta, llamar a uno de los tres agentes encargados de los tratamientos.

Agente Psicologo1

1. CU Tratar Ira

Este será el encargado de dar tratamiento a la ira según las preferencias del estudiante y el tratamiento indicado para revertir dicha emoción.

Agente Psicologo2

1. CU Tratar Miedo

Este será el encargado de dar tratamiento al miedo según las preferencias del estudiante y el tratamiento indicado para revertir dicha emoción.

Agente Psicologo3

1. CU Tratar Tristeza

Este será el encargado de dar tratamiento a la tristeza según las preferencias del estudiante y el tratamiento indicado para revertir dicha emoción.

2.1.4 Identificación de Roles

En este diagrama se exploran las responsabilidades de cada agente a través de escenarios específicos de cada Rol mediante Diagramas de Secuencia. Cada Rol es obtenido componiendo varias tareas en una conducta resultante.

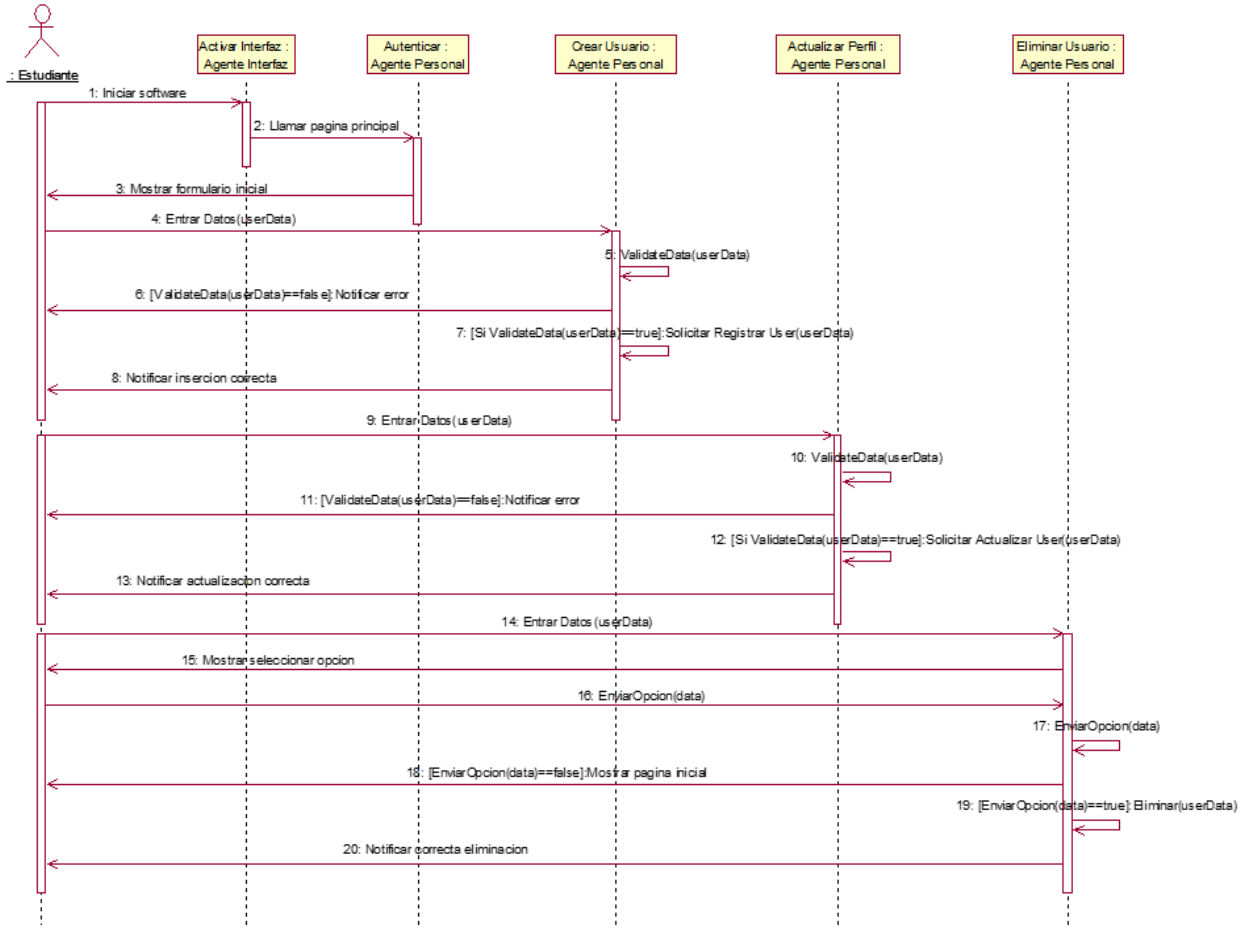

Ilustración 4: Diagrama de Secuencia: Gestionar Usuario

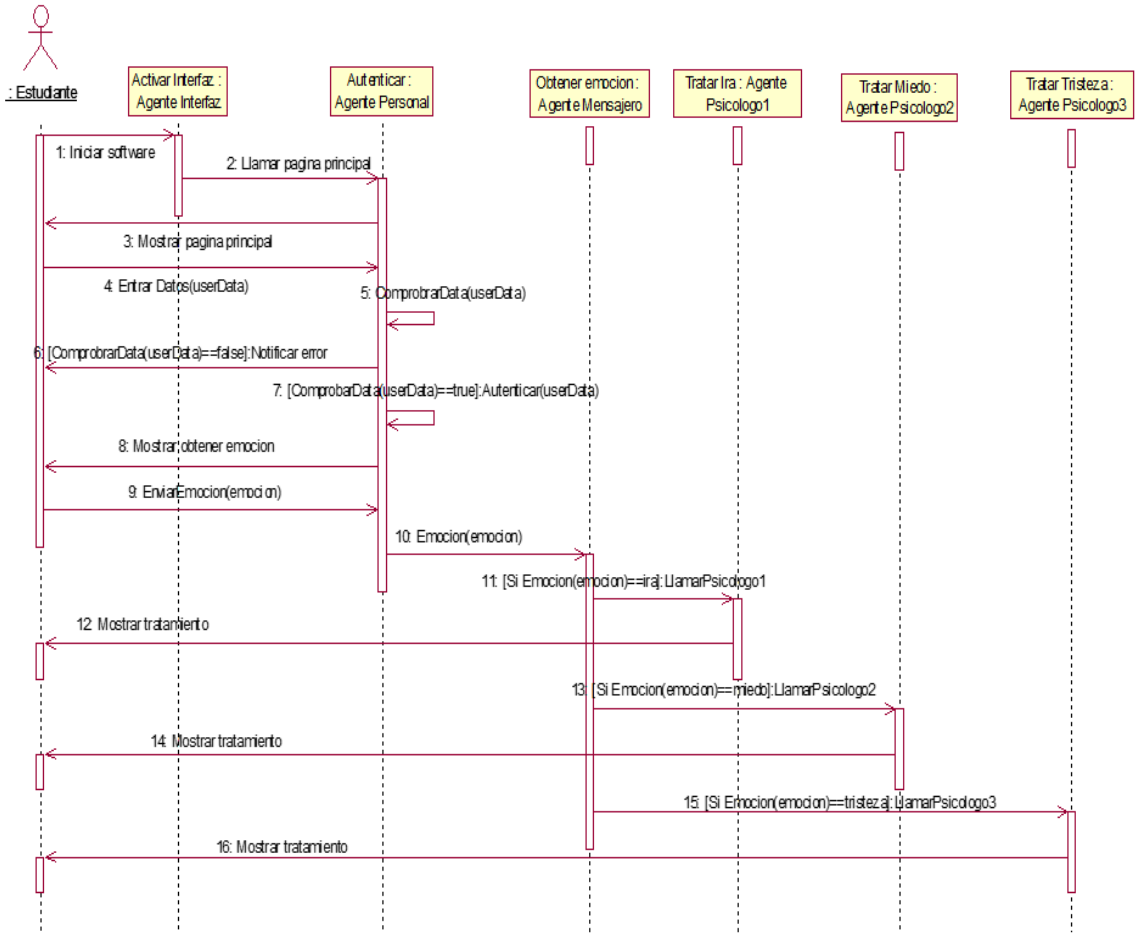

Ilustración 5: Diagrama de Secuencia: Obtener Emoción

Identificación de Roles de los agentes propuestos

Agente Interfaz

1. Rol Activar Interfaz

Agente Personal

1. Rol Autenticar
2. Rol Crear Usuario
3. Rol Actualizar Perfil
4. Rol Eliminar Usuario

Agente Mensajero

1. Obtener emoción

Agente Psicologo1

1. Rol Tratar Ira

Agente Psicologo2

1. Rol Tratar Miedo

Agente Psicologo3

1. Rol Tratar Tristeza

Tabla 7: Gestionar Usuario

Escenario: Crear Usuario	
Descripción	Este escenario muestra el curso normal que sigue un usuario externo para acceder al sistema.
CU Asociados:	CU Crear Usuario. CU Validar Información. CU Recibir Información.
Roles Asociados	**Descripción**
Rol Activar Interfaz **(Agente Interfaz)**	Este rol es el responsable de la comunicación entre los usuarios y el sistema. Su principal actividad es activar

	la interfaz con el agente personal

Tabla 8: Obtener Emoción

Escenario: Obtener Emoción.	
Descripción	Este escenario muestra el curso normal que sigue el sistema al obtener la emoción.
CU Asociados:	1. CU Autentificar 2. CU Obtener Emoción. 3. CU Tratar Ira. 4. CU Tratar Tristeza. 5. CU Tratar Miedo.
Roles Asociados	**Descripción**
Rol Activar Interfaz **(Agente Interfaz)**	Este rol es el responsable de la comunicación entre los usuarios y el sistema. . Su principal actividad es activar la interfaz con el agente personal
Rol Autenticar **(Agente Personal)**	El objetivo de este rol es permitir la entrada de los diversos usuarios registrados al sistema. Sus principales actividades son verificar su usuario y contraseña.
Rol Obtener Emoción **(Agente Mensajero)**	Este rol es el encargado de captar la emoción sentida por el usuario una vez que haya accedido al sistema. Su principal actividad es captar la emoción.
Rol Tratar Ira **(Agente Psicólogo1)**	Este rol es el encargado de tratar la Ira en caso de que esta sea la emoción

	sentida por el estudiante. Su principal actividad es tratar la ira.
Rol Tratar Miedo **(Agente Psicólogo2)**	Este rol es el encargado de tratar el miedo en caso de que esta sea la emoción sentida por el estudiante. Su principal actividad es tratar el miedo.
Rol Tratar Tristeza **(Agente Psicólogo3)**	Este rol es el encargado de tratar la tristeza en caso de que esta sea la emoción sentida por el estudiante. Su principal actividad es tratar la tristeza.

2.1.5 Especificación de Tareas

En este diagrama se especifican a través de Diagramas de Actividades las capacidades de cada agente. Además, permite modelar el ciclo de vida de cada agente, colaboraciones que necesita y comunicaciones en que participa.

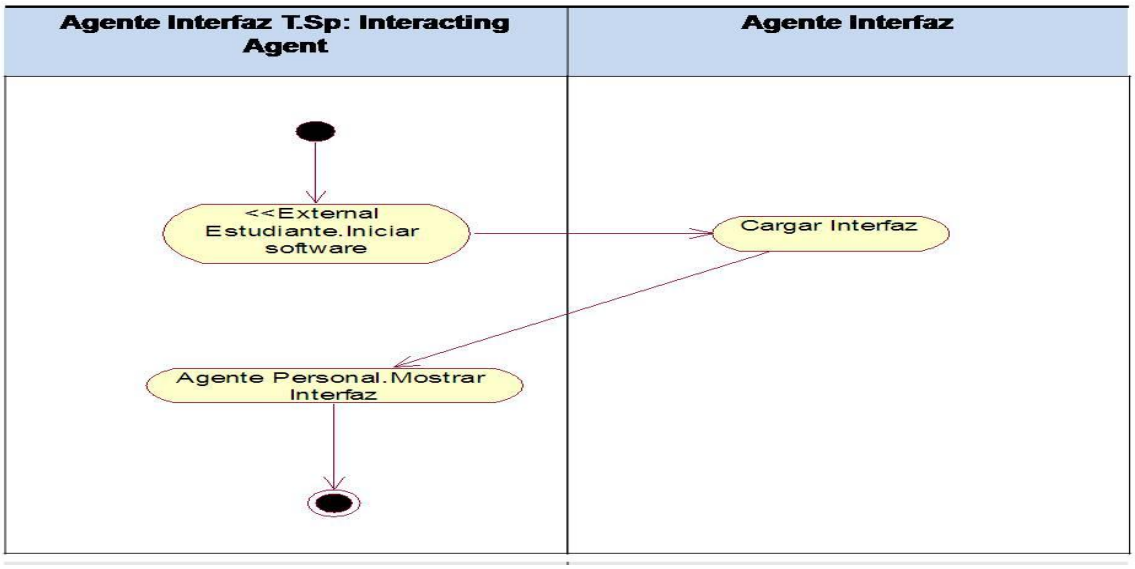

Ilustración 6: Diagrama de Actividades: Interfaz

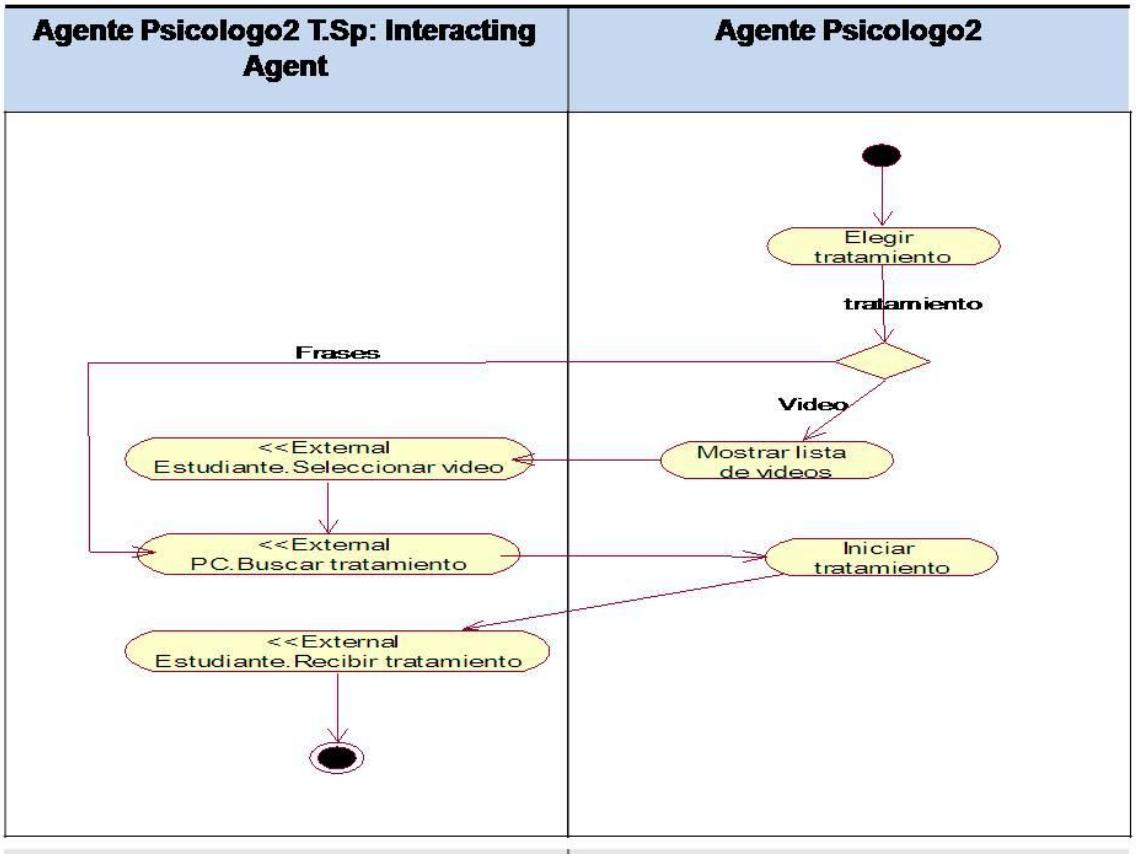

Ilustración 7: Diagrama de Actividades: Psicólogo2

2.2 Modelo de Sociedades de Agentes

2.2.1 Descripción de Ontología

En esta fase se describe el conocimiento atribuido a agentes individuales y a sus interacciones mediante uso de diagramas de clases y restricciones OCL.

En esta fase se pueden encontrar dos diagramas: Descripción de Ontología del Dominio y Descripción de Ontología de Comunicación.

2.2.2 Diagrama de Descripción de Ontología de Dominio

Este diagrama describe la ontología del dominio que representa a las entidades a través de las clases.

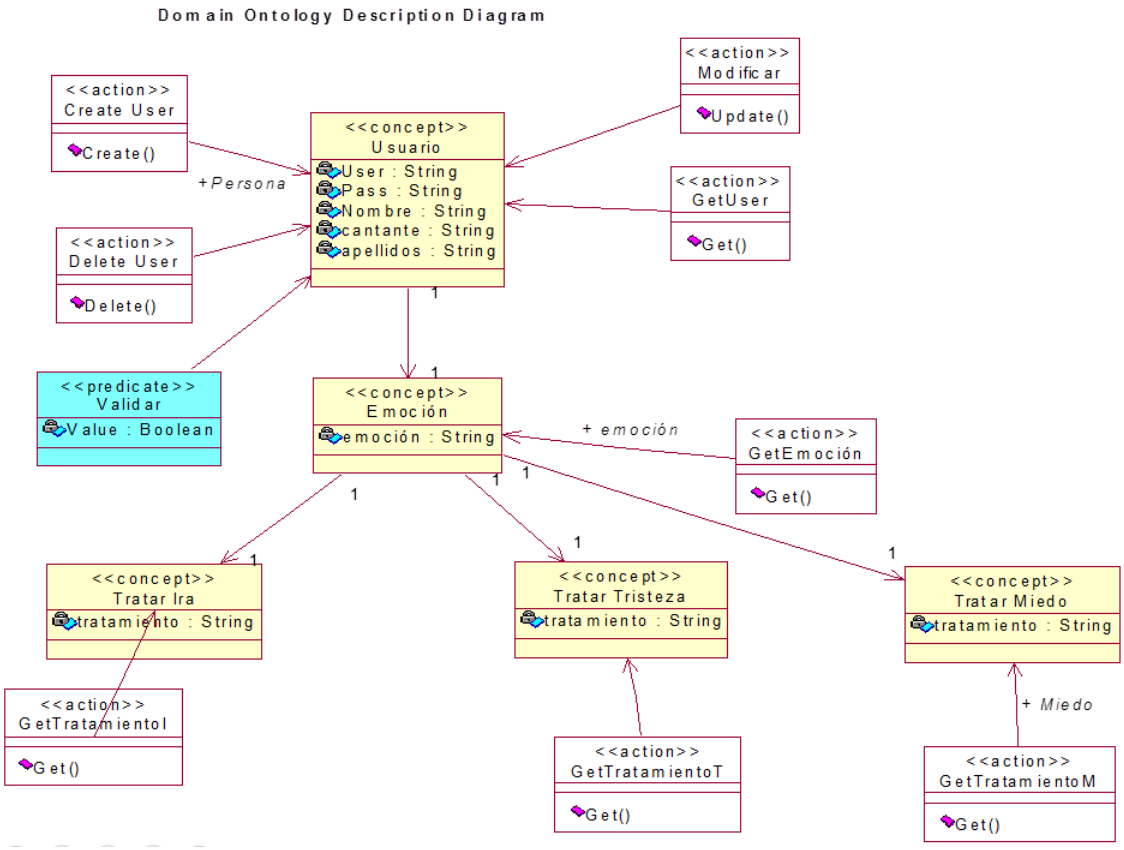

Ilustración 8: Diagrama de Descripción de Ontología del Dominio

2.2.3 Diagrama de Descripción de Ontología de Comunicación

Este diagrama enfoca el conocimiento de los agentes y las relaciones comunicativas entre los mismos.

Para el diseño de este diagrama se parte de los resultados de la fase de Identificación de Agentes. Para cada agente identificado se introduce una clase, así como una asociación para cada comunicación entre los agentes.

Para cada comunicación es imprescindible especificar tres elementos: ontología, lenguaje y protocolo de interacción.

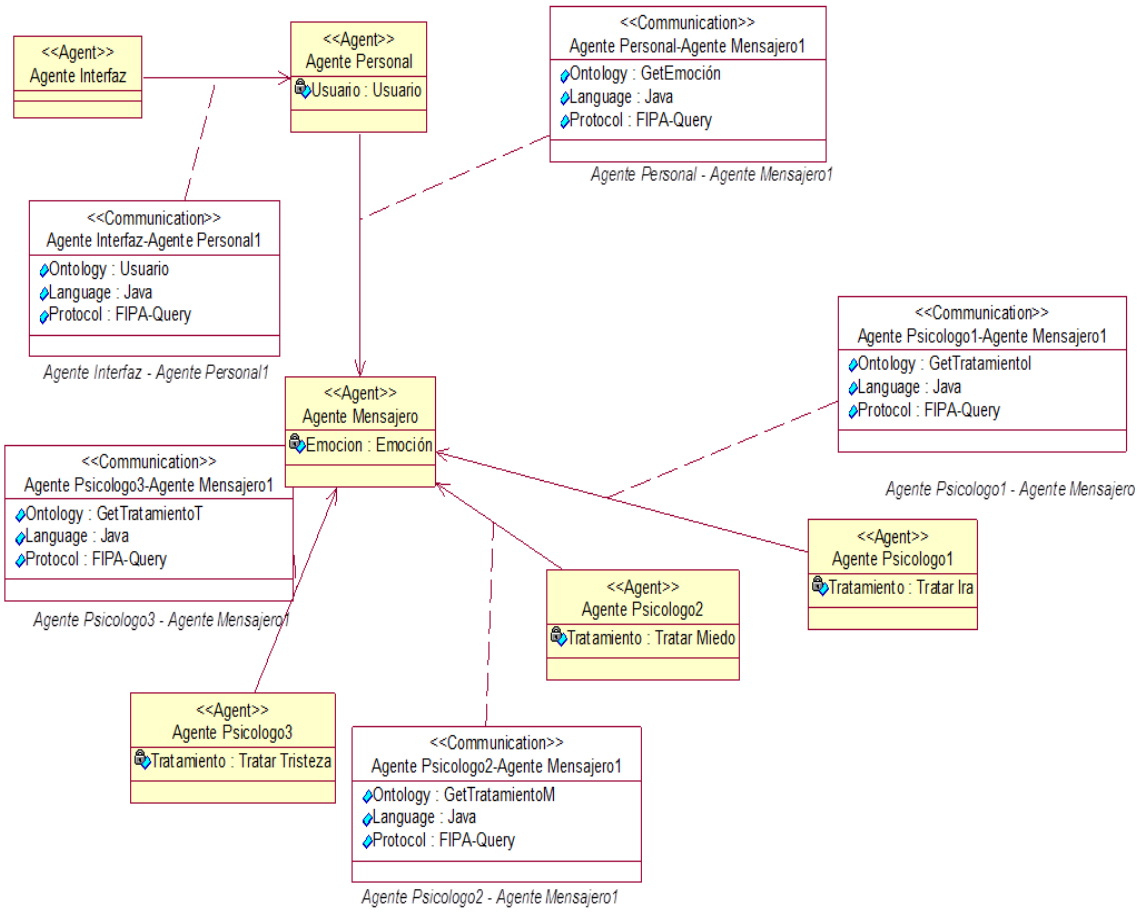

Ilustración 9: Diagrama de Descripción de Ontología del Comunicación

2.3 Descripción de Roles

En esta fase se introducen los agentes como clases y se especifican las tareas de cada uno dentro de los roles definidos anteriormente, además quedan establecidas también las comunicaciones entre cada rol.

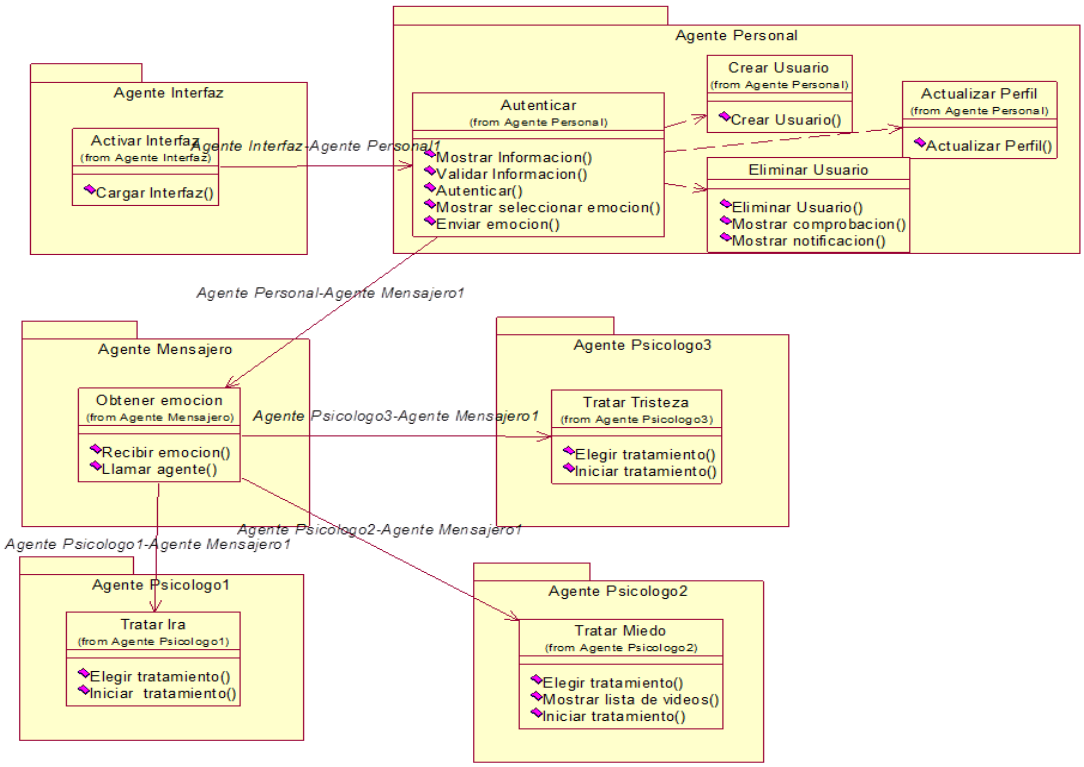

Ilustración 10: Diagrama de Descripción de Roles

Conclusiones

En este capítulo se realizó una detallada descripción del diseño del SMA con el apoyo de la herramienta PASSI. El cual es de vital importancia para la implementación del sistema.

Capítulo 3.Implementación de la solución propuesta con PASSI

En esta tercera fase del diseño realizado utilizando PASSI, se estructura el sistema multiagente así como las especificaciones de cada agente. Esta etapa es de vital importancia pues en ella se definen las relaciones y funcionalidades de cada agente, así como su alcance, lo cual facilitará la etapa de implementación.

3.1 Modelo de Implementación de agente

En esta fase se pueden encontrar dos diagramas mediante los cuales se describe la estructura de las clases de agente.

3.1.1 Definición de estructura del SMA

En este es representado al sistema multiagente como un conjunto de clases, donde es representado cada agente y sus relaciones.

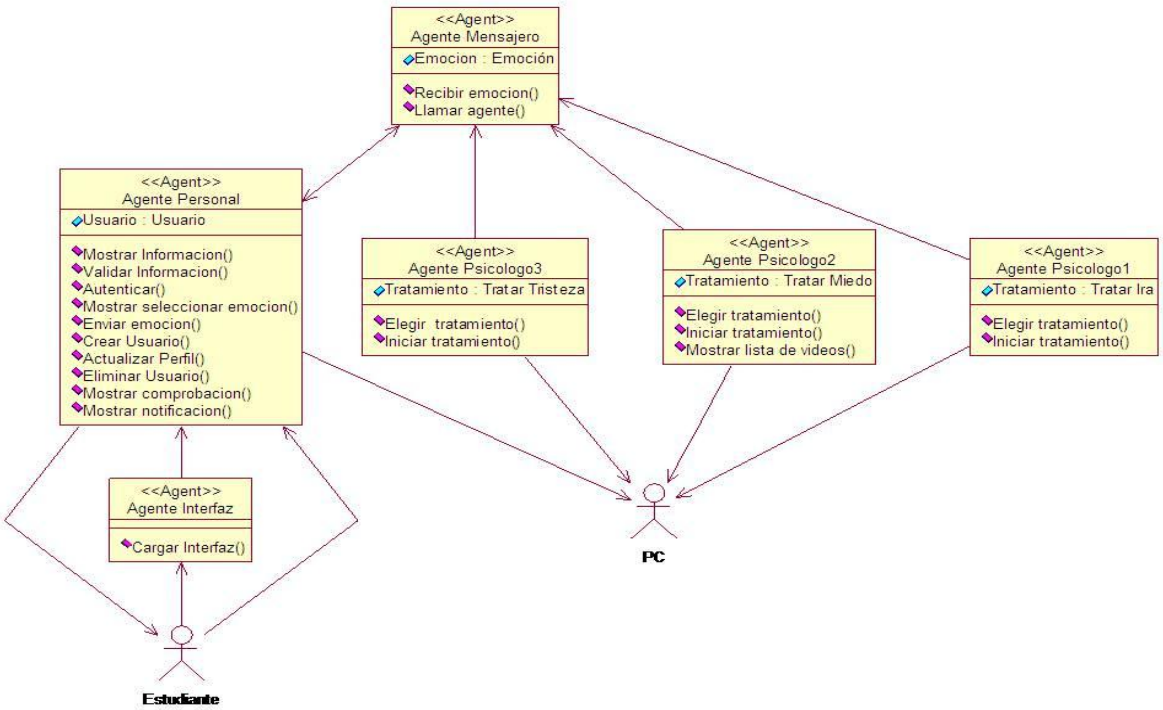

Ilustración 11: Diagrama de Definición de Estructura de un SMA

3.1.2 Definición de estructura del agente simple

En este diagrama se describe cómo quedará estructurado cada agente, sus funciones y la información que contendrá cada uno de estos para su ejecución.

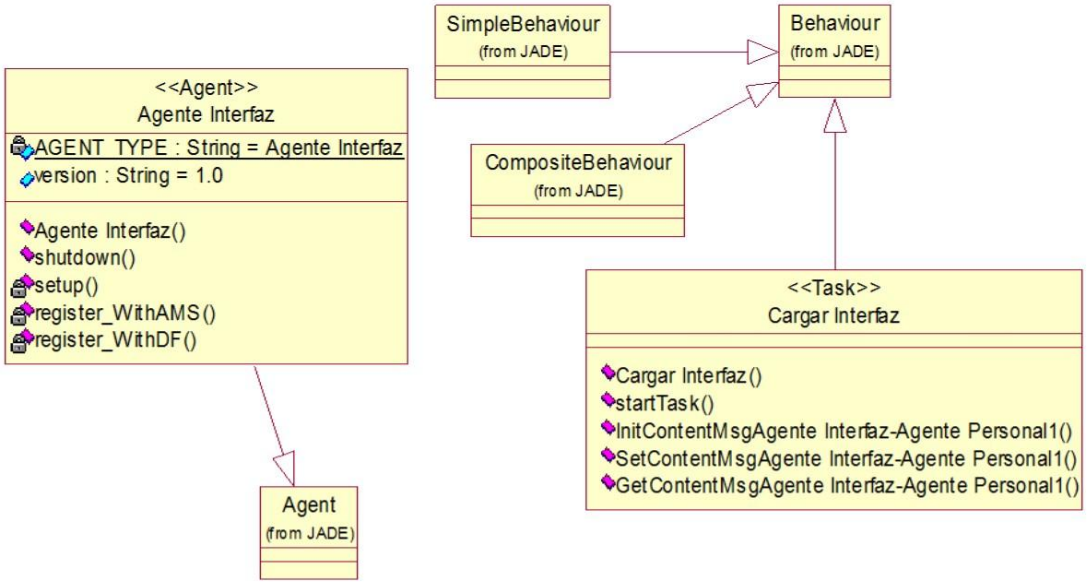

Ilustración 12: Diagrama de Definición de Estructura de un Agente Simple: Agente Interfaz

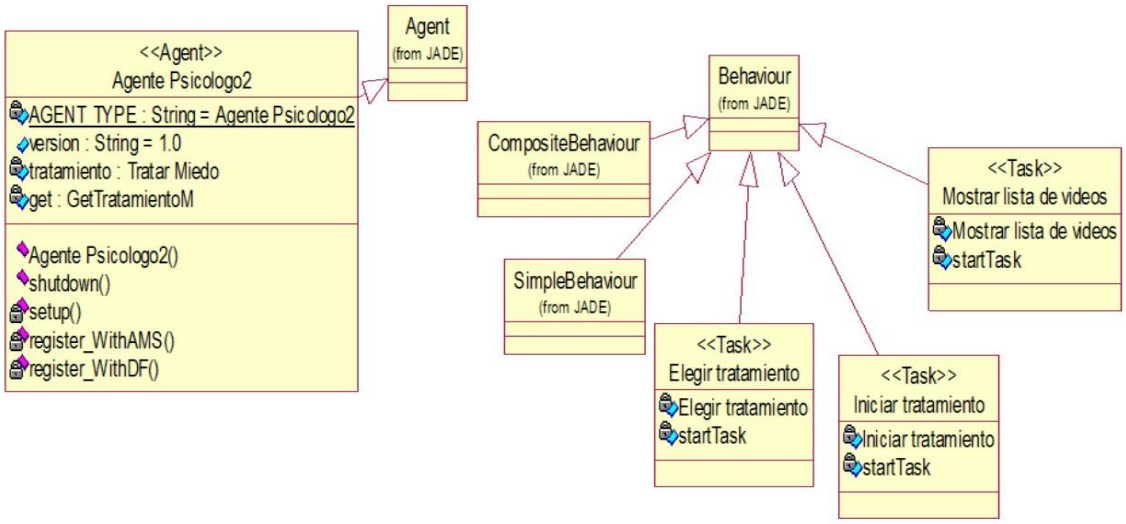

Ilustración 13: Diagrama de Definición de Estructura de un Agente Simple: Agente Psicologo2

3.2 Descripción de conducta del agente

3.2.1 Descripción de conducta del SMA

Ente diagrama se muestra el flujo constante de información entre los agentes, mostrando esta descripción mediante Diagramas de Actividades, donde cada agente constituye una calle.

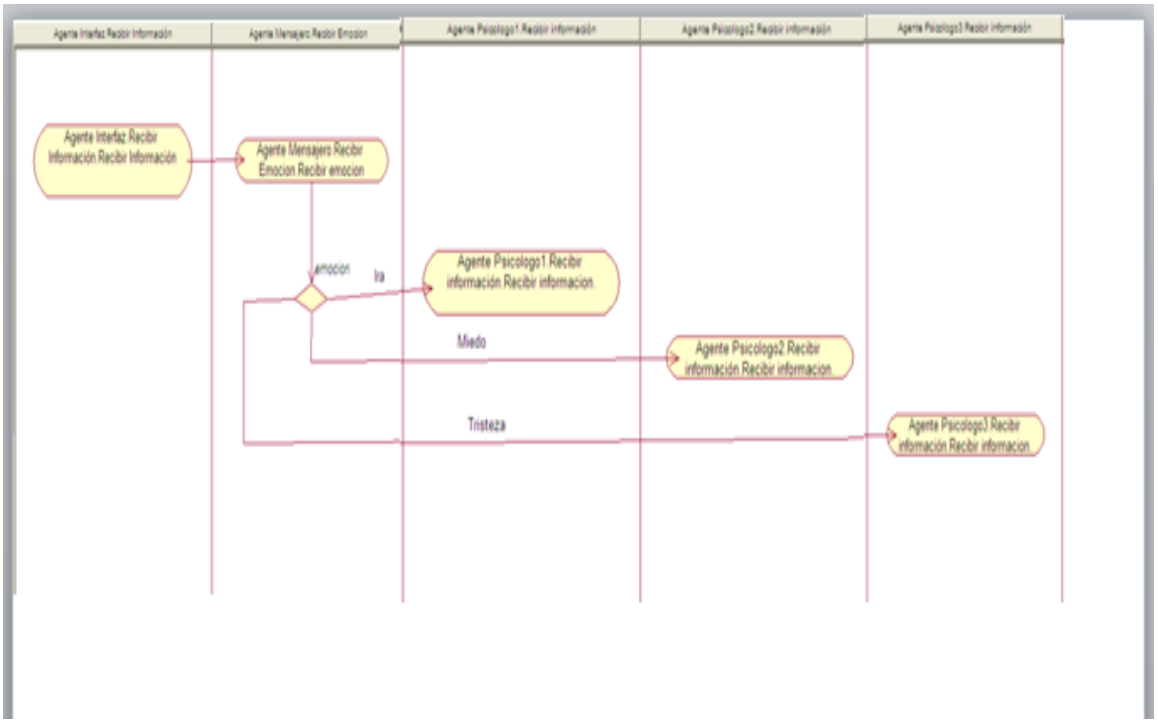

Ilustración 14: Diagrama de Actividad SMA

3.2.2 Descripción de conducta del Agente simple

Está compuesto por la implementación de los métodos introducidos en los Diagramas de Definición de Estructura del Agente, se puede describir de forma libre, ya que es de acuerdo al criterio del diseñador. Esta fase no es objetivo de esta investigación por lo que no fue implementada.

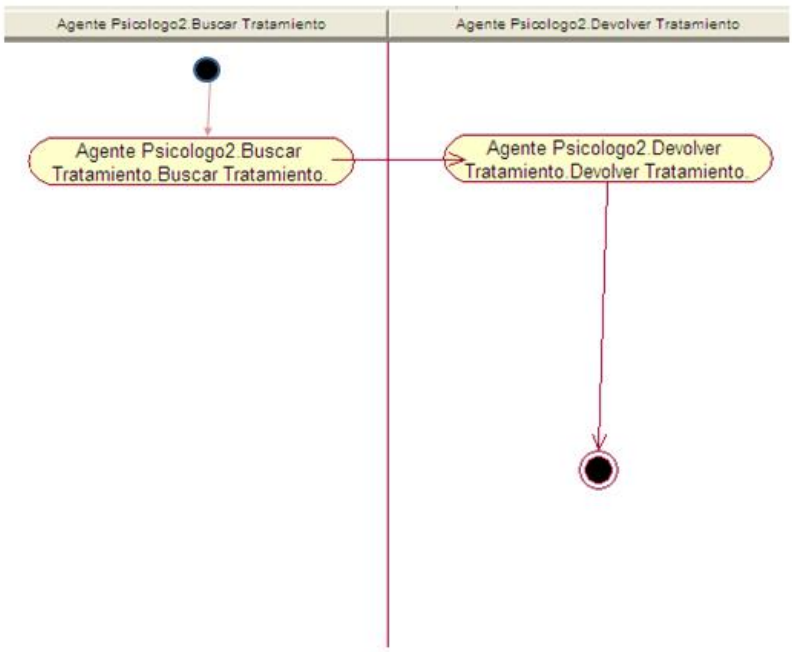

Ilustración 15: Diagrama de Actividad: Psicólogo2

3.3 Modelo de Código

3.3.1 Biblioteca de código reutilizable

A partir del diseño de todas las fases anteriores con la herramienta PTK, se genera una biblioteca de código reutilizable automáticamente. Este código facilita la implementación de las interfaces de cada agente.

- **Código Agente Interfaz:**

```
public class Agente Interfaz extends Agent {

  private static String AGENT_TYPE = Agente Interfaz;
  public String version = 1.0;
  private Usuario usuario ;
  private Emoción emocion ;
```

```
  private Modificar update ;
  private GetUser getUser ;
  private Create User createUser ;
  private Delete User delete ;
  private GetTratamiento getTratamiento ;
  private GetEmoción getEmocion ;
  private Tratar Ira tratarIra ;
  private Tratar Tristeza tratarTristeza ;
  private Tratar Miedo tratarMiedo ;
  private Validar validar ;

  public void Agente Interfaz(String platform, String name, String ownership) {
    //insert here your code
  }

  public void shutdown() {
    //insert here your code
  }

  private void setup() {
    //insert here your code
  }
```

- **Código Agente Psicólogo2**

```
public class Agente Psicologo2 extends Agent {

  private static String AGENT_TYPE = Agente Psicologo2;
  public String version = 1.0;
```

```
private Emoción emocion ;
private Tratar Miedo  miedo ;
private GetTratamiento getTratamiento ;
private GetEmoción getEmocion ;
private GetTratamientoM getTratamientoM ;

public void Agente Psicologo2(String platform, String name, String ownership) {
  //insert here your code
 }

public void shutdown() {
  //insert here your code
 }

private void setup() {
  //insert here your code
 }
```

3.4 Modelo de Despliegue

En este diagrama se constituye la ubicación de cada agente en todas las unidades de procesamiento.

3.4.1 Configuración de Despliegue

Aquí se detalla la posición de los agentes en las unidades físicas presentes.

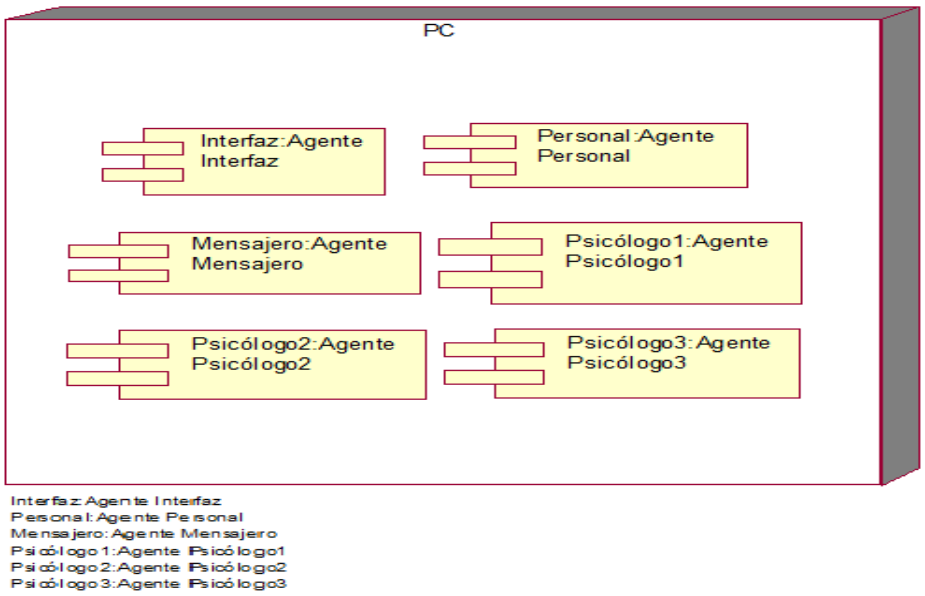

Ilustración 16: Diagrama del Modelo de Despliegue

1.5 Interfaces del SMA

Finalizada la fase del diseño utilizando la metodología PASSI, se realizó la implementación del SMA con el apoyo de la plataforma de implementación de agentes JADE, middle-ware que integrado con el IDE Netbeans facilitó el proceso de construcción del sistema.

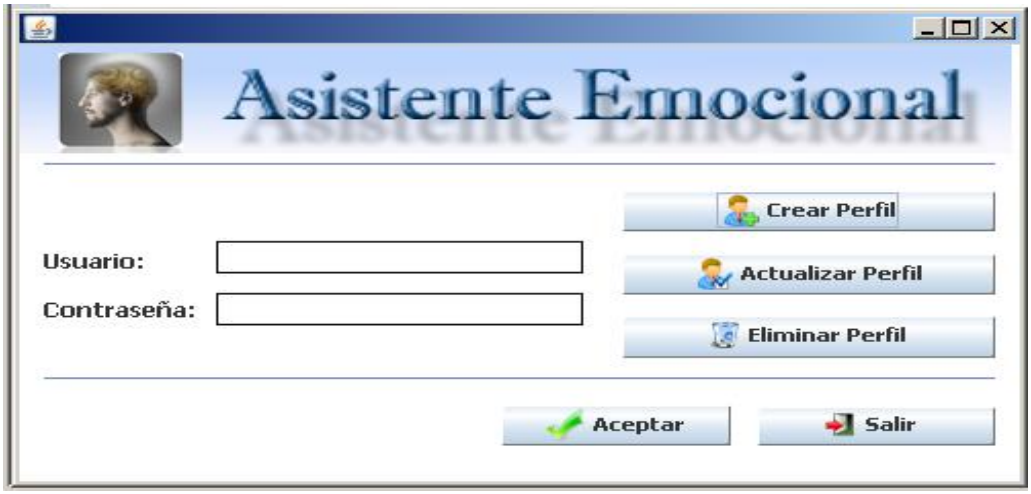

Ilustración 17: Asistente Emocional: Gestión de Usuario

Asistente Emocional será el nombre del sistema y de la pantalla de loging, en la cual el estudiante tendrá las opciones de:

- Crear Perfil: Dará acceso a otra ventana con datos necesarios del estudiante para convertirse en usuario del sistema.
- Actualizar Perfil: Dará acceso a una ventana similar donde el estudiante podrá modificar cualquier dato que haya cambiado excepto el usuario.
- Eliminar Perfil: Mostrará una pequeña ventana emergente en la cual el estudiante tendrá la opción de ser eliminado de forma permanente del sistema.

Una vez creado el perfil con su usuario y contraseña el estudiante podrá tener acceso al sistema.

Ilustración 18: Asistente Emocional: Obtener Emoción

Luego de logeado, esta ventana será la encargada de capturar la emoción que el estudiante pueda sentir en ese momento, una vez obtenida la emoción, el sistema

procederá a despertar al Agente Psicólogo2 el cual será el encargado de revertir la emoción miedo.

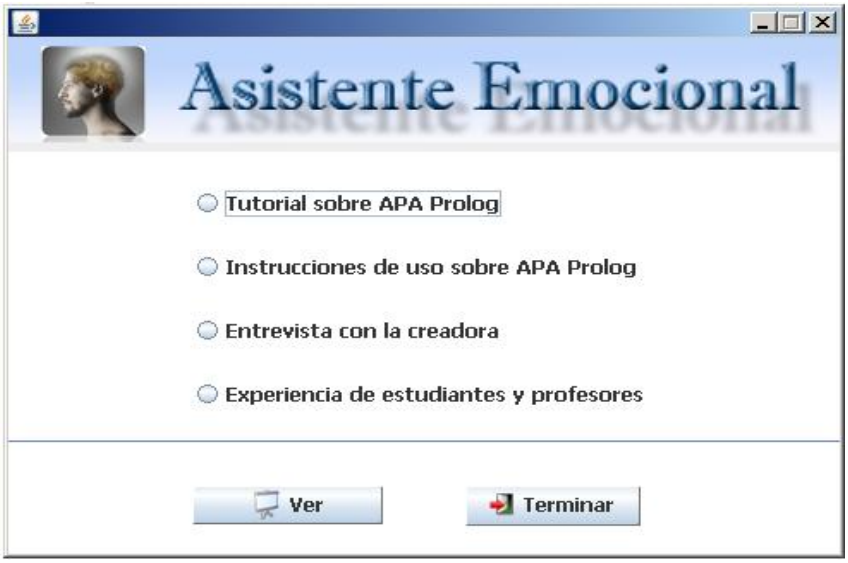

Ilustración 19: Asistente Emocional: Seleccionar Video

Para la emoción miedo existen dos tratamientos que serán aplicados al estudiante de forma aleatoria, si el tratamiento aplicado es video, en esta ventana emergente el estudiante podrá escoger entre cuatro opciones las cuales lo guiarán en el uso de esta herramienta.

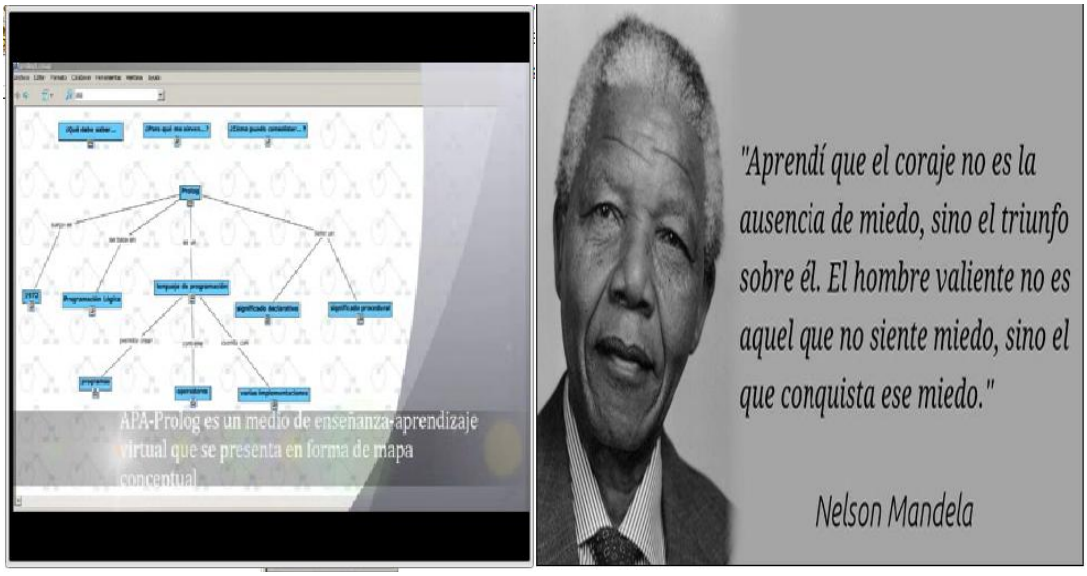

Ilustración 20: Asistente Emocional: Tratamientos Miedo

Estos son los tratamientos que se aplicarán a los estudiantes en caso de que presente la emoción miedo.

Conclusiones

En este capítulo se realizó una descripción detallada de cada agente así como las partes que constituyen el SMA. Para esto se tuvo en cuenta sus funcionalidades de cada agente, los datos necesarios para los mismos, la secuencia de pasos para la comunicación entre estos y la ubicación permanente de cada agente implementado. Además, se obtuvo una biblioteca reutilizable de código vital para la implementación del SMA.

Conclusiones

1. El estudio de los fundamentos teóricos metodológicos permitió contextualizar las diferentes variables que intervienen en el problema, lo cual posibilitó esclarecer que los agentes a utilizar en el SMA, no eran agentes inteligentes.
2. Posibilitó apreciar el ilimitado campo de la programación afectiva, así como su importancia en el manejo de emociones, sobre todo en aplicaciones informáticas para la enseñanza.
3. Se estudiaron varias metodologías pero PASSI se consideró la más adecuada pues abarca todo el ciclo de vida del SMA, lo cual contribuyó una guía efectiva para el desarrollo de la misma.

Recomendaciones

1. Establecer un orden de prioridad en los tratamientos.
2. Realiza una versión android para satisfacer el auge de dispositivos con este Sistema Operativo.

Bibliografía

Martínez, D. I. T. (2009). Arquitectura Multiagente para Entornos de Inteligencia Artificial. (Tesis Doctoral), Universidad de Salamanca.

Sosa, E. C. y. A. (2007). La Computación Afectiva y el Arte Interactivo.

Aguiar, E. (2006). Pruebas de Sqlite en un sistema Linux

guia para iniciarse.

Ahogado, D., Reinemer, A. M., & González, E. (2003). AOPOA: Aproximación Organizacional para Programación Orientada a Agentes.

Escofet, C. M. El lenguaje SQL.

García, A. E. (2003). Manual Práctico de SQL.

Greco, C., Morelato, G., & Ison, M. Emociones Positivas: Una herramienta psicológica para promocionar el proceso de resiliencia infantil.

Guevara, J. M. L. d. (*Fundamentos de Programación en Java*

Hípola, P., & Vargas-Quesada, B. (1999). Agentes Inteligentes: definición y tipología.

Jiménez, M. L. V. (2006). Emociones Positivas. Retrieved from Papeles del Psicólogo website: file:///C:/Documents%20and%20Settings/YoelB/Escritorio/College/Tesis/capitulo1/emociones%20positivas/Papeles%20del%20Psic%C3%B3logo.htm

Marchetti, T. J., & García, A. J. Metodologías de desarrollo de sistemas multi-agente:

un análisis comparativo.

Marchetti, T. J., & García, A. J. Plataformas para Desarrollo de Sistemas Multiagente. Un Análisis Comparativo.

Peñaranda, J. C. (2015). *Las Promesas de la Computación Afectiva.*

Rodríguez, J. A. P. (2009). Emociones negativas y su impacto en la salud mental y física.

Sanz, J. J. G. (2003). Metodologías para el desarrollo de sistemas multi-agente. *Revista Iberoamericana de Inteligencia Artificial*.

Zunino, A., Berdún, L., & Amandi, A. (2001). JavaLog: un Lenguaje para la Programación de Agentes. *Revista Iberoamericana de Inteligencia Artificial*.

Baldassarri, S. Computación Afectiva.

H4dm. (2014). from file:///C:/Documents%20and%20Settings/YoelB/Escritorio/College/Tesis/revisar/Computaci%C3%B3n%20Afectiva/%C2%BFQu%C3%A9%20es%20la%20computaci%C3%B3n%20afectiva.htm

Marchetti, T. J., & García, A. J. Plataformas para Desarrollo de Sistemas Multiagente. Un Análisis Comparativo.

Anexos

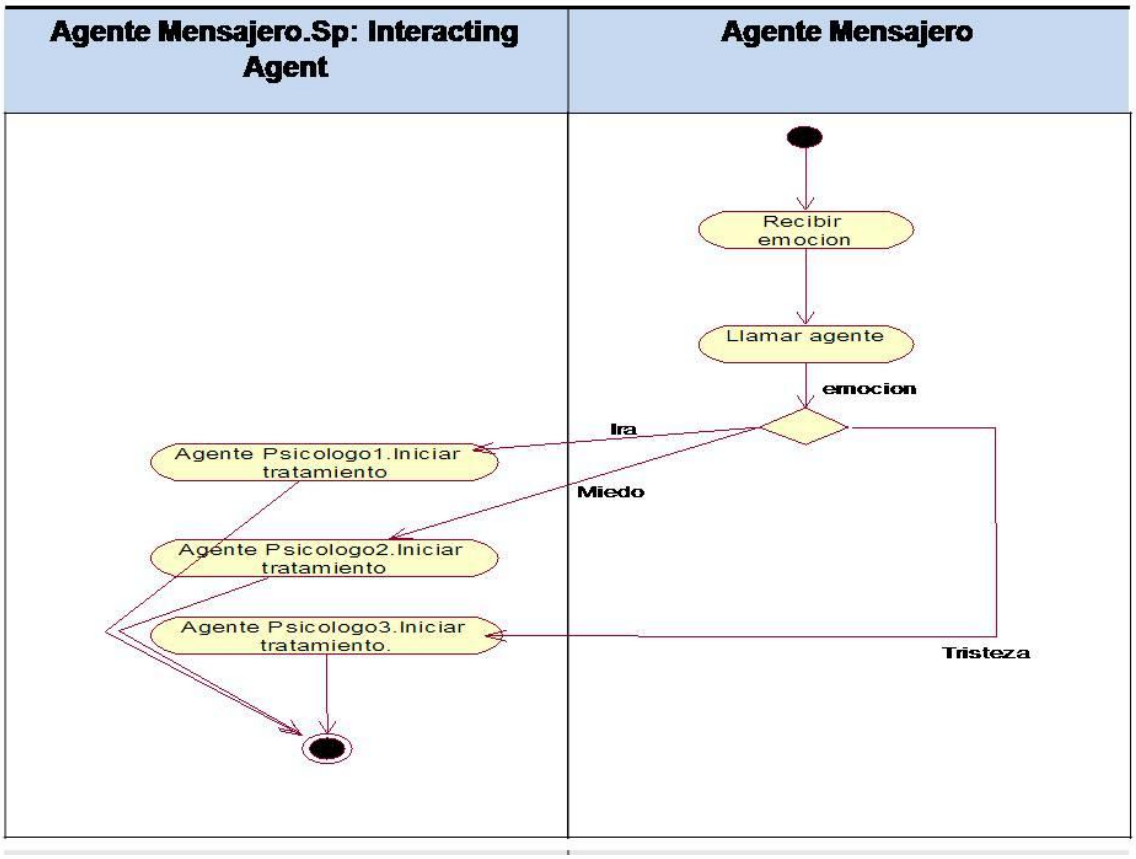

Ilustración 21: Diagrama de Actividades: Agente Mensajero

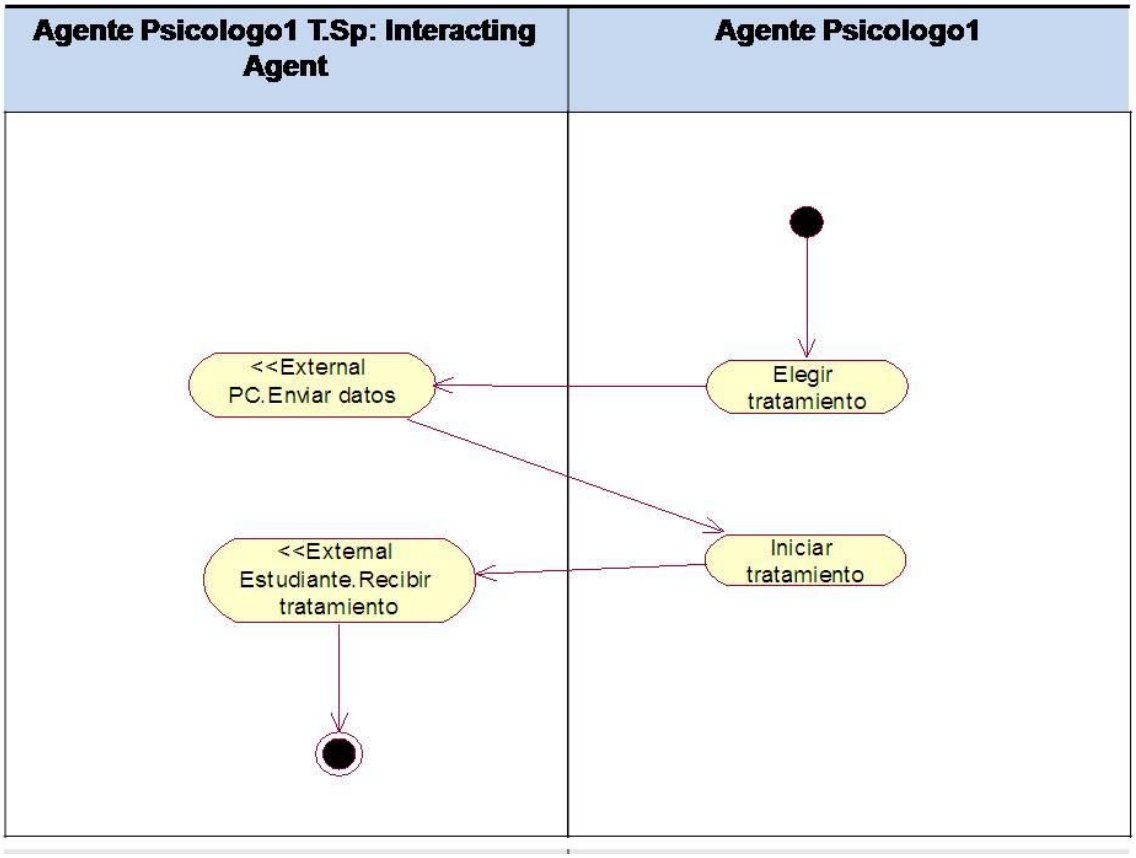

Ilustración 22: Diagrama de Actividades: Agente Psicólogo1

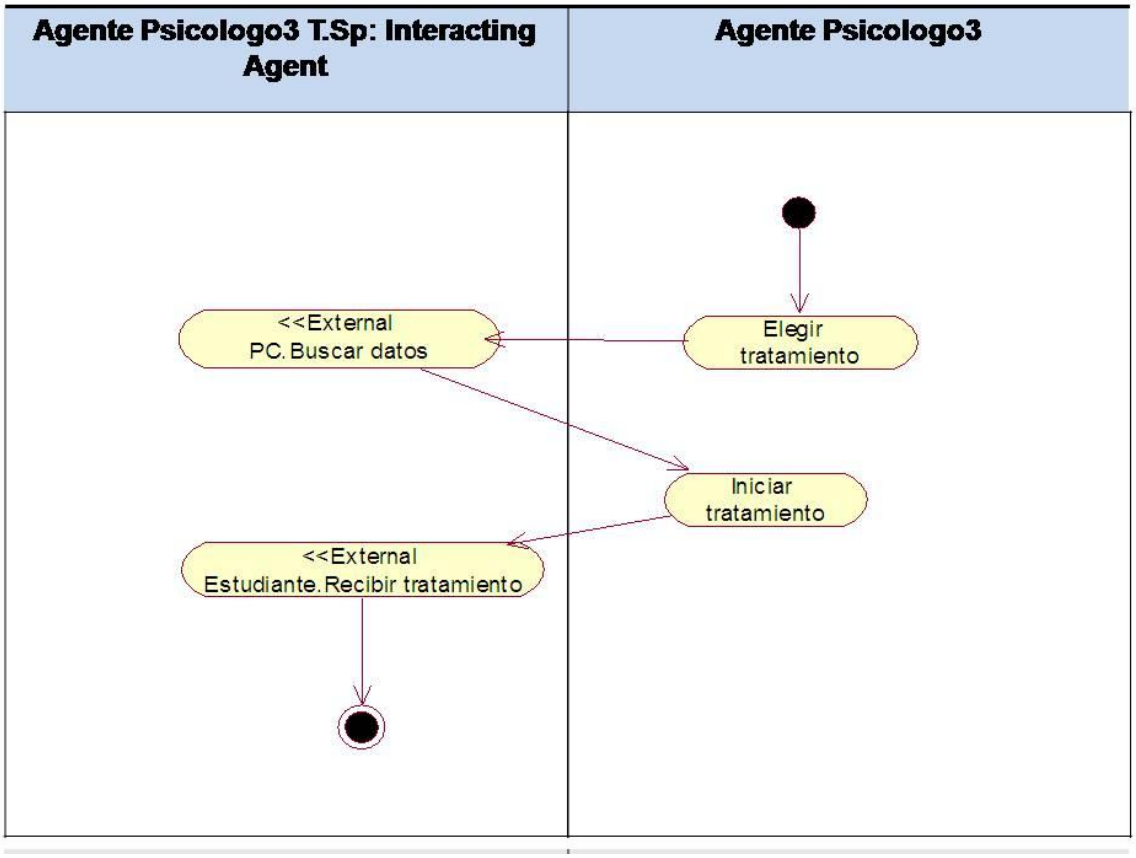

Ilustración 23: Diagrama de Actividades: Agente Psicólogo3

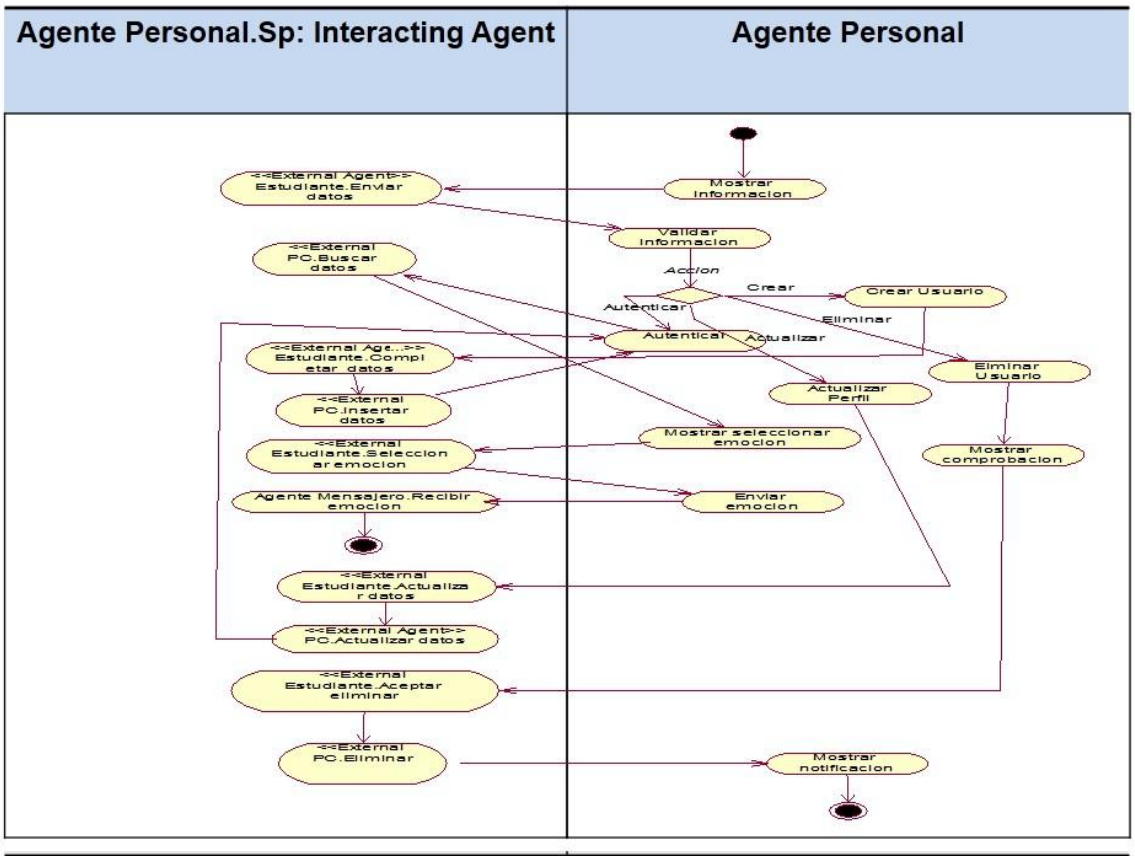

Ilustración 24: Diagrama de Actividades: Agente Personal

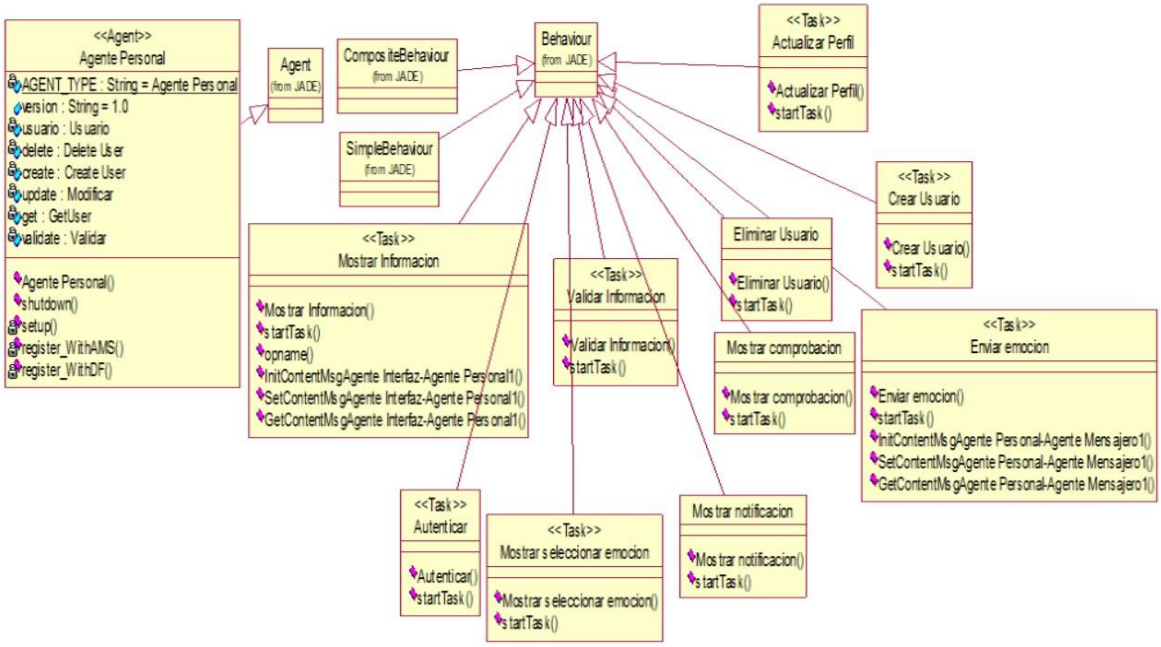

Ilustración 25: Diagrama de Definición de Estructura de un Agente Simple: Personal

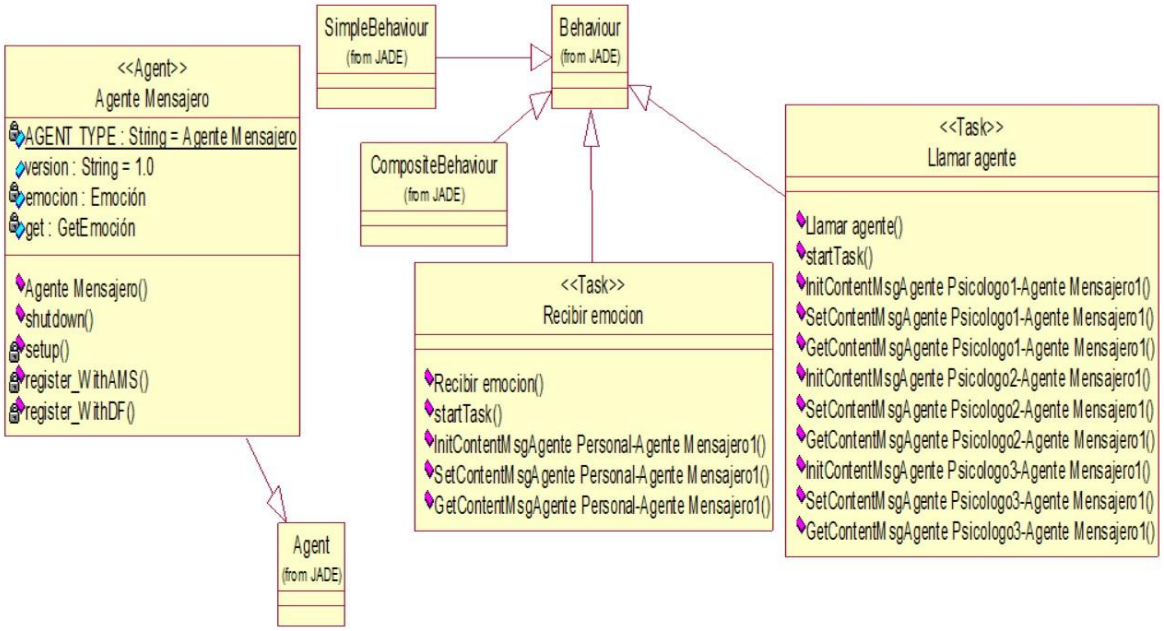

Ilustración 26: Diagrama de Definición de Estructura de un Agente Simple: Mensajero

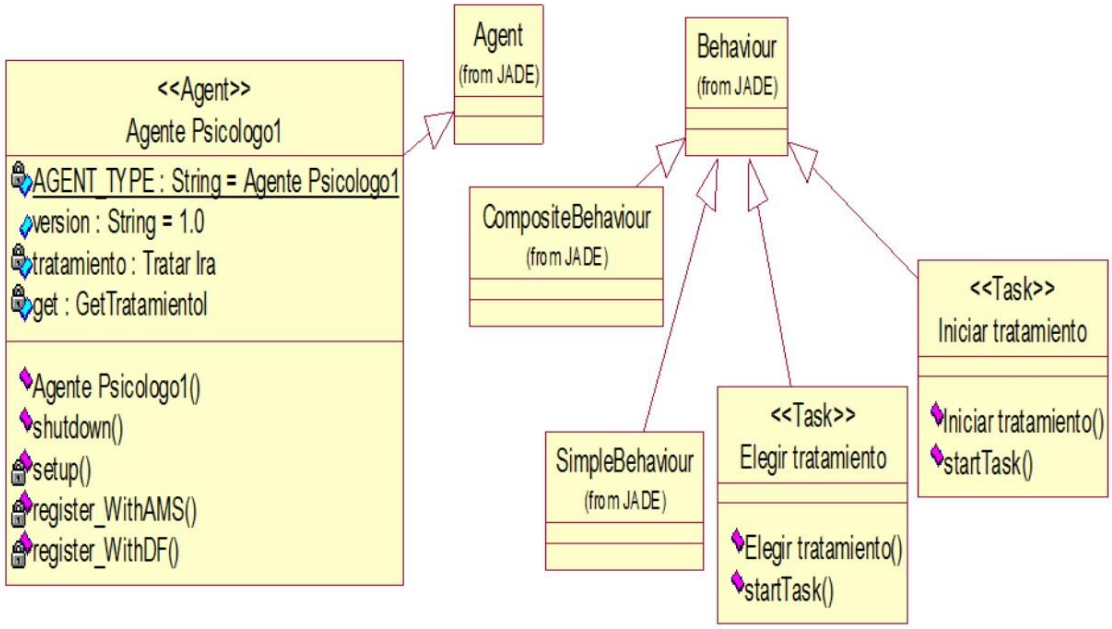

Ilustración 27: Diagrama de Definición de Estructura de un Agente Simple: Psicólogo1

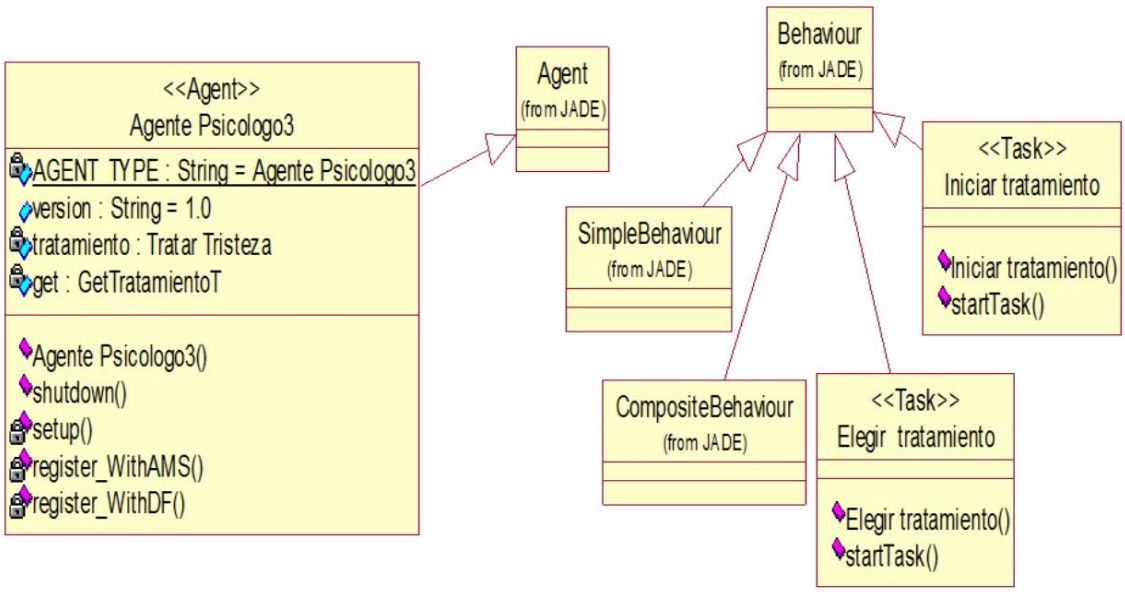

Ilustración 28: Diagrama de Definición de Estructura de un Agente Simple: Psicólogo3