UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

TESIS DE GRADO

"CUALIFICACIÓN DE CONOCIMIENTOS MEDIANTE AGENTE INTELIGENTE APLICADO AL CAMPO DE LA INFORMATICA"

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA

MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS

POSTULANTE: GARY GERARDO MAMANI CONDORI

TUTOR METODOLOGICO: M.Sc. ALDO RAMIRO VALDEZ ALVARADO

ASESOR: M.Sc. CARLOS MULLISACA CHOQUE

LA PAZ – BOLIVIA

2017

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

DEDICATORIA

A Dios por mandar personas que creyeron en mí en momentos cruciales de mi vida…

Al padre German Z.E.P.D.

Mi padre Gerardo Mamani Z.E.P.D.

"los sueños son el principio de las metas realizadas…"

AGRADECIMIENTOS

A mis padres por las palabras de aliento.

Al Lic. Aldo Valdez por toda la ayuda y orientación para la realización de este trabajo.

A mis amigos: Alvaro, Adrian, David, Jose, Marcelo, William, Hugo, Israel, Vladimir, Nayra, Daniela, Milenka, Eymi, Melina por la ayuda sincera en momentos determinantes de mi vida.

A Marcelo Berazain por ser ese compañero de tesis y estar en los momentos más difíciles. Realmente un buen amigo.

A mis docentes por tener paciencia y dedicación con mi persona durante mis años de estudio universitario.

A mis hermanos Vladimir y Efrain. Esta es la prueba que vinimos a este mundo a triunfar…

RESUMEN

El presente trabajo de tesis de grado titulado “Cualificación de conocimientos mediante agente inteligente aplicado al campo de la informática”, muestra el desarrollo del modelo de un sistema multi-agente para lograr cualificar conocimientos de estudiantes egresados de la carrera de Informática de la UMSA.

Las metodologías de sistemas multi-agente Prometheus e INGENIAS forman parte fundamental del trabajo puesto que ambas metodologías unificadas brindan un modelado del sistema de manera más detallada aprovechando de las virtudes que nos ofrecen ambas metodologías.

Puesto que el principal objetivo es alcanzar la cualificación de conocimientos en estudiantes se hará uso de reactivos de opción múltiple en la elaboración de preguntas con un contexto en algunos casos de situaciones reales de trabajo para poder obtener las cualidades de los estudiantes sometidos al test.

ABSTRACT

This thesis work entitled "Qualification of knowledge through intelligent agent applied to the field of computer science", shows the development of the model of a multi-agent system to achieve qualification knowledge of students graduated from the career of Computing of the UMSA.

The methodologies of multi-agent systems Prometheus and INGENIAS are a fundamental part of the work since both unified methodologies provide a more detailed modeling of the system taking advantage of the virtues offered by both methodologies.

Since the main objective is to achieve the qualification of knowledge in students, multiple choice reagents will be used in the elaboration of questions with a context in some cases of real work situations in order to obtain the qualities of the students submitted to the test.

ÍNDICE Pág.

MENÚ DE TABLAS Pág.

ÍNDICE DE FIGURAS Pág.

CAPÍTULO I

MARCO REFERENCIAL

1.1. Introducción

La cualificación tiene hoy en día un importante lugar en el ámbito laboral, pues esta puede ser capaz de mostrar conocimientos y cualidades en cuanto a una formación profesional, esta acción es tan dinámica que un individuo debe seguir capacitándose para seguir cualificado de lo contrario sus conocimientos podrían quedar obsoletos.

La educación superior tuvo muchos cambios en sus procesos de evaluación se puede apreciar muchos métodos para medir el nivel de conocimiento obtenido a lo largo de un curso o una materia impartida en un ciclo de enseñanza aprendizaje por lo cual se vale algunas herramientas o medios para poder responder a esta problemática y poder medir de una u otra manera las capacidades de estudiantes o profesionales en diferentes áreas, este tipo de pruebas no fue la excepción en su evolución con respecto a la tecnología pues se fueron evolucionando e innovando las maneras de obtener estos resultados requeridos al momento de cuantificar el nivel de conocimiento al respecto.

Teniendo en cuenta que hoy en día se hace más frecuente la intensión de automatizar procesos que son cotidianos y no solo de automatizarlos si no de mejorarlos y hacer que estas tareas se realicen de manera autónoma, es en este punto que se ve la necesidad de implementar Inteligencia Artificial (IA) en las aplicaciones y sistemas que hoy en día se usan.

Aclarando el punto anterior dentro de lo que es la Inteligencia Artificial se ve la intervención de agentes inteligentes en este aspecto lo cuales tienen la función de actuar en un entorno y responder a los estímulos adaptándose a los cambios del entorno aprendiendo de la experiencia, tomando decisiones correctas de acuerdo al procesamiento finito de la

información que este maneja viendo así una gran ventaja a la hora de realizar tareas determinadas.

Viendo estos puntos se ve la posibilidad de automatizar y facilitar el procedimiento de evaluación en un test en el cual se requiere cualificar los conocimientos de los estudiantes o profesionales para distintos fines.

En la tesis propuesta se tiene la intensión de cualificar los conocimientos en el campo de la informática por medio de un test y este a su vez este administrado por un agente inteligente.

1.2. Antecedentes

La escala de Binet-Simón (publicada en 1905) es la gran madre de los test, con numerosos hijos en todas las latitudes del planeta, y especialmente en los Estados Unidos de América.

Sin embargo, desde siempre el ser humano ha anhelado medirlo todo. Incluso el espíritu. De hecho, en la Grecia clásica los filósofos se enzarzaron en enconadas disputas acerca de la posibilidad de conocer los límites del alma. Heráclito sentenció: "nunca podrás medir los límites del alma, tan grandes son sus confines". No obstante, en el siglo VI a. de C. los pitagóricos, no conformes con esta advertencia, intentaron identificar el alma con un número.

Posteriormente, la aspiración de cuantificarlo todo estuvo aletargada durante muchos siglos. Hasta finales del siglo XVIII no contamos con un hecho decisivo para la constitución de la psicometría como ciencia matemático-experimental: el astrónomo del rey Jorge III de Inglaterra había observado que su ayudante obtenía unas determinaciones del paso de los astros por el retículo del telescopio que diferían hasta ocho décimas de segundo de las suyas; creyendo que se debía a su ineptitud, despidió a su ayudante. Algunos años después otro astrónomo, conocedor de este hecho, realizó una serie de experimentos que le evidenciaron la existencia de una ecuación personal, es decir, que el **tiempo psíquico** varía

de una a otra persona (de ahí la divergencia en las mediciones astronómicas). Este descubrimiento puso sobre la mesa de los filósofos primero y de los psicólogos después un tema de investigación sobre lo que luego se llamaría tiempo de reacción: la constatación objetiva de las diferencias individuales.

En la Universidad de Cambridge explicaba filosofía y psicología a finales del siglo XIX sir FRANCIS GALTON, que es el padre de la psicometría. Su laboratorio antropométrico le dio fama universal, y en 1884, con ocasión de un Congreso Internacional de Higiene, invitó a los londinenses a someterse, por el módico precio de tres peniques, a ciertas mediciones, como agudeza visual, tiempo de reacción, capacidad de discriminación táctil, etc. Resultado: un año después publicó la primera tabla de baremos, con la que los visitantes del laboratorio comparaban sus resultados.

La revista Mind publicó en 1890 un artículo *Mental test and measurements* de uno de los discípulos predilectos de Galton, James Mc Keen Cattell, en el que apareció por primera vez el término test. Puede decirse, pues, que la psicometría nació oficialmente ese año. (Mind 1890)

Desde mediados de la década de los 70's, los índices insosteniblemente altos de paro estructural y la transformación de los procesos productivos gracias a las TIC (Tecnologías de la Información y Comunicación) han llevado el concepto de cualificación (*skill*) y los procesos que permiten obtenerlas hasta el centro del debate sobre las políticas de empleo idóneas para modernizar una economía. Para esclarecer los conceptos de cualificación se debe empezar admitiendo la construcción social de esta. Como ha señalado Word (1981), "para un sociólogo, toda cualificación es una construcción social, ya que ninguna procede de tecnologías caídas del cielo". (Rigby & Sanchis, 2006)

Los debates sobre el concepto de cualificación profesional suelen ceñirse a su exclusiva dimensión técnica o profesional, a capacidades de manipulación y conocimientos sobre el proceso de trabajo, desarrollados por formación o experiencia. (Rigby & Sanchis, 2006)

Entre 1890 y 1905 los psicómetras utilizaron unos test basados en que “no existe nada en la inteligencia que no se halle antes en los sentidos” y pretendieron medir la inteligencia con pruebas de agudeza visual. La Segunda Guerra Mundial supuso un considerable impulso a la investigación experimental sobre los test psicotécnicos en Estados Unidos, especialmente en el Ejército del Aire: la necesidad de seleccionar masivamente entre millones de candidatos se sumó a la necesidad cualitativa (exigencias desconocidas hasta entonces). Se aplicaron a millones de reclutas de ambos bandos pruebas psicotécnicas (en Estados Unidos el *Army General Clasification* Test fue respondido por cerca de diez millones). Además, el grupo de investigaciones psicológicas de la U.S. *Air Force* desarrolló notablemente el análisis factorial. Terminada la Gran Guerra, se generalizó la aplicación de pruebas y test psicotécnicos, tanto en la escuela como en la clínica psiquiátrica y en la industria. Desde entonces, los test psicotécnicos han experimentado un gran desarrollo, especialmente en su aplicación al psicodiagnóstico clínico y a la selección de personal. (Ángel Fernández Muñoz.Psicólogo, Pedagogo, Máster en Dirección de Recursos Humanos. Profesor del CEF y de Universidad Complutense.)

El estudio de agentes tuvo sus comienzos en trabajos de conductistas psicológicos como *Skinner* en 1953 los cuales trataban de reducir la psicología a correlaciones de entrada y salida. Con lo cual se llegó a la metáfora de la computadora de los agentes de Putman en 1960 y Lewis en 1966, así es como poco a poco se fue avanzando en el desarrollo de esta metodología hasta que en 1983 Jon Doyle propuso que el diseño de agentes racionales sea la meta o uno de los propósitos principales a conseguir en la inteligencia artificial. Esta propuesta fue tomada en consideración, pero no consolidada, poco después en 1988 Horvitz fue más específico al sugerir que el empleo de la racionalidad brindaría la máxima utilidad como fundamento de la inteligencia artificial. (Choque, 2013)

Así fue como se declaró los parámetros mínimos de racionalidad en que una entidad sea considerada un agente, en 1991 Russell y Wefald abordaron explícitamente las posibles arquitecturas de los agentes.

Finalmente se llega a la conclusión o definición que un agente es una entidad capaz de percibir su entorno, de procesar la información (percepciones) recogida y determina la salida o respuesta racional que efectuara, es decir que analiza cual es la mejor acción, teniendo en cuenta la maximización del resultado, la eficiencia y la eficacia. Existen muchas clasificaciones entre los agentes inteligentes según su arquitectura y según su comportamiento, pero en la que se basará esta será en los agentes de razonamiento. (Choque, 2013)

A continuación, se muestran algunos trabajos relacionados con el tema de estudio.

Sistema inteligente "Diagnóstico médico" es un sistema de diagnóstico médico aplicando casos probabilísticos para la detección de enfermedades. (Heckerman, 1991)

Tesis de grado "Evaluación del coeficiente intelectual a personas de 12 a 65 años mediante agentes inteligentes". Esta tesis se centra en introducir el concepto de agente inteligente en el campo de la psicología empleado en el desarrollo de un test de RAVEN. (Tarqui, 2009)

Tesis doctoral "Modelo Adaptativo Multi-Agente para la Planificación y Ejecución de Cursos Virtuales Personalizados" Escuela de Sistemas Universidad Nacional de Colombia, Medellín. Esta tesis se propuso como objetivo enfrentar algunas limitaciones encontradas en el referencial teórico respecto a deficiencias en el aprovechamiento de tecnologías de punta en los sistemas de educación virtual, en particular la falta de un esquema gen érico de personalización de los cursos, lo que requiere definir los elementos que deben ser tenidos en cuenta en cada estudiante (modelo del estudiante) para adaptar el curso y a la vez asociar estas diferencias con materiales y actividades que reconozcan en la práctica al estudiante (modelo de dominio).

El resultado final es la definición e implementación de un modelo genérico de cursos adaptativos, neutro, abierto, funcional, intercambiable que privilegia al estudiante en el proceso educativo. El modelo propuesto es neutral ante las diferentes visiones y enfoques

desde lo pedagógico y tecnológico y facilita que al momento de la implantación se tomen las decisiones respectivas con respecto a taxonomías, características a incluir y elementos determinantes en el proceso de adaptación, mientras que se mantenga la consistencia entre estos atributos y las reglas que guían la personalización. La inclusión de un novedoso pre-planificador da gran versatilidad al sistema y es uno de los aportes importantes de esta tesis, pues permite entregar en forma transparente el problema de la generación del curso como un problema de planificación en IA (*AI Planning*), para ser resuelto mediante SHOP2, un potente algoritmo HTN. (Mendez, 2009)

1.3. Planteamiento del problema

Según estudios realizados por Sally Brown en su libro Evaluar en la Universidad, menciona las falencias de los exámenes escritos tradicionales proponiendo el cambio de este hacia otro tipo de evaluaciones, revelando así que la efectividad de cada examen escrito tradicional hoy en día carece de efectividad calidad y profundidad, en los cuales intervienen diferentes factores y que según el autor se escribió en diferentes casos de investigación. También asevera que los exámenes escritos tradicionales no reflejan el aprendizaje de los estudiantes y afirmando que existen estudios realizados en Nueva Zelanda con bastantes evidencias de que los docentes no son los mejores haciendo exámenes válidos, lo cual nos induce a experimentar otros tipos de pruebas para lograr resultados más precisos y confiables. (Brown,2003)

Los estudiantes de la carrera de Informática de la universidad Mayor de San Andrés que poseen ciertos conocimientos en diferentes áreas del campo de la informática desconocen el grado de profundidad de los mismos puesto que no se tiene un parámetro al cual regirse, teniendo una diseño curricular vigente desde 1995 en la cual no se contemplan los requerimientos de un mercado laboral actual, de ser expuesto este grado de conocimiento perfeccionarían su especialidad y más aún el hecho de también saber las falencias en otras

áreas y poder convertir estas en fortalezas, no solo para el estudiante también mismo modo para la carrera de informática tomar las medidas necesarias en la curricula ya mencionada.

Al intentar realizar un test como una de las herramientas para poder cualificar al estudiante nos topamos con la intervención de muchos factores, como ser la ideología del individuo que crea el test y una supuesta desventaja al no cubrir preguntas de todas las áreas dentro de un campo y lo mismo en el caso que este sea realizado por un grupo de docentes que no personalizan un test para extraer la mayor información posible de un estudiante para poder evaluarlo, el no tener un banco de preguntas amplio y variado en niveles y áreas que en la actualidad fueron incrementándose con el avance de la tecnología y la coyuntura social, de alguna manera imposibilita los medios para poder adaptarse al estudiante, limitando el aprovechamiento en la extracción de conocimientos en otras áreas que son bastas en campo de la informática.

1.3.1. Problema

En la actualidad existe dificultad al querer cualificar el nivel de conocimiento que el estudiante posee para ingresar al mercado laboral, que parámetros tomar como referencia para este fin, y tener un método mediante el cual se pueda extraer de manera minuciosa los conocimientos que este posee teniendo en cuenta que un estudiante en su particularidad es diferente. Y darle a conocer el nivel de sus fortalezas en las áreas dentro del campo y al mismo tiempo las falencias que se tiene para así tener una visión más amplia y poder tomar las medidas pertinentes ante este veredicto, tanto institucionalmente a la carrera como a nivel personal en el estudiante.

Se identificaron además los siguientes problemas:

- Herramientas para medir las cualidades y conocimientos en el campo de la informática difícilmente encontrada en el medio, lo que ocasiona realizar test escritos.

- Test generales no adaptables al conocimiento que posee el evaluado.
- Dificultad en la extracción de conocimiento del estudiante para una cualificación más eficaz.
- Estudiantes que ignoran los parámetros de nivel con respecto a sus conocimientos
- Las referencias de requerimientos laborales estandarizados a nivel local aplicables en alguna prueba de cualidades no se realizan.

Lo cual lleva a realizar la siguiente pregunta:

¿Cómo cualificar los conocimientos de un estudiante egresado de la carrera de Informática?

1.4. Definición de Objetivos

1.4.1. Objetivo General

Desarrollar un sistema multi-agente que cualifique los conocimientos de estudiantes egresados de la carrera de informática.

1.4.2. Objetivos Específicos

Desarrollar un sistema multi-agente para la cualificación de conocimientos en informática.

- Aplicar niveles de preguntas en el test para determinar el nivel de conocimiento en el ámbito de la informática.
- Aplicar métricas de estándares internacionales para la elaboración de las preguntas del test.
- Aplicar reglas para la elaboración de las preguntas del test de tal manera que estas cumplan con los objetivos propuestos
- Analizar los posibles casos en los cuales el agente inteligente interactúa con las respuestas del estudiante en el sistema.
- Aplicar parámetros reales en el test acorde a las necesidades del mercado laboral

- Modelar las entidades, acciones y reacciones dentro del sistema para alcanzar un nivel fiable de cualificación

1.5 Hipótesis

El uso de un sistema multi-agente para la cualificación de conocimientos en estudiantes egresados de la carrera de Informática determina una aceptación al resultado de manera positiva en al menos un 70%.

1.6. Justificación

La tesis surge de un problema real entre los estudiantes que poseen conocimientos y cualidades en diferentes áreas del campo de la informática, muchos de ellos al no saber lo que se requiere en el ámbito laboral y dado el nivel en el que se encuentran, se topan con problemas tanto personales como labórales, y también a la empresa para la cual trabajan y a la institución de educación superior a la que representan, se beneficiarían al conocer lo mucho o poco que les falta para perfeccionar o mejorar las cualidades y conocimientos que poseen ante un mercado laboral al cual saldrán y obtener un panorama más amplio de a lo que se afrontaran, y al mismo tiempo seguir mejorando la imagen que tienen los profesionales pertenecientes a la Universidad Mayor de San Andrés.

1.7. Alcances y Limites

1.7.1. Alcance

El alcance del presente documento se centrará en el modelado de los agentes inteligentes que intervendrán en la cualificación de conocimientos, esta acción la realizara mediante un test teniendo en cuenta la premisa de tener métricas con las cuales poder realizar de mejor manera el presente trabajo "Pautas Curriculares para Programas de Licenciatura en Informática" publicado el 20 de Diciembre del año 2016 por la *Association for Computing Machinery (ACM)* y la *IEEE Computer Society*.

El sistema multi-agente que administrará el test tendrá las siguientes acciones:

- Controlará el nivel de preguntas que brindará al usuario
- Evaluará las respuestas que el usuario realice.
- Brindará un resultado con la resolución del test revelando el nivel de conocimiento conseguido en cada área del presente caso
- La base de preguntas tendrá contenidos de redes y comunicación solamente.
- El prototipo simulara las actividades encomendadas en el modelo exclusivamente para el caso de cualificación.

1.7.2. Límites

El sistema multi-agente tendrá las siguientes limitantes:

- En el caso del presente sistema multi-agente solo se abarcará el área de telemática.
- El sistema no brindara una puntuación cuantificable de los conocimientos de los estudiantes de la carrera de informática.
- Las preguntas del test serán elaboradas de acuerdo al contenido mencionado pero las creaciones de estas no serán elaboradas por un equipo profesional como se aconseja.
- El sistema multi-agente no tiene contemplado brindar estadísticas de resultados de la población que se someta al test.

1.8. Metodología

1.8.1. Metodología de Investigación

El método utilizado para el presente trabajo es el método científico el cual es un proceso de etapas, que obtiene conocimiento valido: no es subjetivo, es verificable, es reducible, es reusable, es factible.

Sus etapas cumplidas son: Observación la cual debe ser sistemática que tiene que ver con el problema, hipótesis la cual es una suposición que forma parte de una posible solución, experimentación, teoría y ley.

El objetivo de este estudio consiste en llegar a conocer las situaciones, costumbres y actitudes predominantes a través de la descripción exacta de las actividades, objetos y personas. Su meta no se limita a la recolección de datos, sino a la predicción e identificación de las relaciones que existen entre dos o más variables. Con el fin de extraer generalizaciones significativas que contribuyan al conocimiento.

1.8.2. Metodología de Ingeniería

La Ingeniería de Software Orientada a Agentes (AOSE), al igual que la ingeniería del software tradicional, el éxito en la aplicación de la ingeniería del software orientada a agentes está relacionada con la correcta aplicación de una metodología y un proceso de desarrollo. Esta correcta aplicación requiere que la metodología empleada se haya aprendido apropiadamente. Sin embargo, si se repasan los contenidos de la principal escuela de verano de agentes, la *European Agent Systems Summer School* (EASS, 2000, 2008, 2009, 2010, 2011, 2012), se encuentran pocas referencias a las metodologías orientadas a agentes y a su aprendizaje. La literatura especializada no es diferente, por lo que, más que hablar de metodologías o tecnologías poco efectivas, quizás haya que hablar de fallos más fundamentales en la transmisión de conocimientos a los que deben aplicar AOSE. En algunos casos como la metodología Prometheus, se reconocen las dificultades para construir este tipo de sistemas, pero los autores se limitan a afirmar que fueron conscientes de ellas en el momento de concepción de la metodología y a apuntar que su metodología ha supuesto una mejora de la situación (Padgham and Winikof, 2005).

Prometheus es una metodología orientada a agentes que nace con el propósito de ser práctica y de propósito general. Con esta premisa, se decide también a acceder a sistemas con agentes basados en objetivos y planes, y se compromete a proporcionar una serie de

herramientas que faciliten el desarrollo a largo plazo. Cuando se define como práctica, quiere decir que tiene como objetivo atender las necesidades de un amplio espectro: desde las personas que se inician en los sistemas multi-agente hasta las necesidades que demanda la industria. Basándose en una definición de agente como la siguiente: una entidad software autónoma, situada en un entorno, reactiva a los cambios en dicho entorno, proactiva en la persecución de objetivos y con habilidades sociales, siendo además flexible y robusta. Prometheus identifica los siguientes conceptos: un agente se relaciona con su entorno mediante sensores y acciones, será proactivo si persigue objetivos, será reactivo si reacciona a eventos, tendrán planes y beliefs, y se comunicarán mediante mensajes que responden a determinados protocolos de interacción. Con estos términos, pueden definirse tanto agentes que se ajusten al modelo de Beliefs *Desires Intentions* (BDI) como agentes que no. El proceso que propone Prometheus se descompone en tres fases: especificación del sistema, diseño arquitectónico y diseño detallado. (Padgham and Winikoff, 2005; Winikoff and Padgham, 2004)

La metodología INGENIAS es una metodología que guía el desarrollo de sistemas multi-agente desde el análisis hasta la implementación. INGENIAS sigue un proceso de desarrollo similar al proceso unificado, definiendo un conjunto de actividades que guían el proceso de elaboración. En INGENIAS, el sistema se modela según diferentes vistas que se especifican a través de diferentes metamodelos, que se complementan con algunos diagramas UML, y en su conjunto representan la totalidad del sistema. Los metamodelos con los que cuenta INGENIAS se instancian en vistas y estos son: modelo de organización, modelo de agente, modelo de interacción, modelo de entorno, y modelo de tareas y objetivos. El modelo de organización, diseña la estructura del sistema, definiendo un espacio donde los agentes, sus recursos y sus tareas/objetivos conviven, además de cómo se relacionan. El modelo de organización, divide estos elementos considerando la definición de grupos y workflows, y define su funcionalidad mediante las tareas y los objetivos. El modelo de agente diseña el comportamiento de agentes individuales en función de sus roles, tareas, objetivos y su estado mental. Con estas herramientas se modela la

funcionalidad del agente, donde además se podrán definir estados mentales determinados para una situación específica, para lo que se cuenta con diversas entidades para describir estados, como hechos y planes. El modelo de interacción define cómo tienen lugar las interacciones, qué actores toman parte en ellas y qué información se intercambia, además del objetivo que persiguen y el contexto al que pertenece. Las interacciones tendrán una especificación asociada, que podrá definirse a partir de diagramas de diferente procedencia (UML, AUML, que define las interacciones a partir de unidades de interacción. El modelo de objetivos y tareas define la especificación de las tareas, así como su relación con los objetivos. Al especificar las tareas, se define qué información produce/consume y qué consecuencias puede tener respecto a los objetivos a los que está relacionada o al estado mental del agente. El modelo de entorno define la integración del sistema con otras aplicaciones u otros agentes. También define los recursos del sistema. Los agentes modelados siguiendo INGENIAS siguen el principio de racionalidad, y la generación de código elige JADE como plataforma objetivo. Al igual que Prometheus, INGENIAS tampoco considera a los agentes móviles. INGENIAS cuenta con la herramienta INGENIAS Development Kit (IDK) como soporte para el desarrollo siguiendo la metodología. Esta herramienta permite diseñar las vistas del sistema siguiendo los modelos definidos por la metodología, a la que añade soporte para validación y verificación. También se incluyen diferentes diagramas UML que contribuyen al desarrollo dirigido por modelos. Además, el IDK cuenta con generación automática de código, lo que le convierte en una herramienta muy completa. Sin embargo, esta herramienta no está completa y no permite incluir toda la semántica de la metodología, lo que supone una desventaja. Una característica negativa de INGENIAS es que, al igual que Prometheus, no considera el modelado de agentes móviles. (Pavón and Gómez-Sanz, 2003; Pavon et al., 2005).

CAPÍTULO II

MARCO TEÓRICO

2.1. Inteligencia Artificial

La IA es la rama de la ciencia que se encarga del estudio de la inteligencia en elementos artificiales y, desde el punto de vista de la ingeniería, propone la creación de elementos que posean un comportamiento inteligente. Dicho de otra forma, la IA pretende construir sistemas y máquinas que presenten un comportamiento que, si fuera llevado a cabo por una persona, se diría que es inteligente. El aprendizaje, la capacidad de adaptación a entornos cambiantes, la creatividad, etc., son facetas que usualmente se relacionan con el comportamiento inteligente. Además, la IA es muy interdisciplinar, y en ella intervienen disciplinas tan variadas como la Neurociencia, la Psicología, las Tecnologías de la Información, la Ciencia Cognitiva, la Física, las Matemáticas, etc.

Se considera que el origen de la IA se remonta a los intentos del hombre desde la antigüedad por incrementar sus potencialidades físicas e intelectuales, creando artefactos con automatismos y simulando la forma y las habilidades de los seres humanos. La primera referencia escrita a un sistema artificial inteligente ocurre en la mitología griega. En ella se atribuye a Hefestos, dios del fuego y las forjas, la fabricación de "dos sirvientes de oro macizo y con inteligencia en su mente y capacidad de hablar". En la Edad Media, San Alberto Magno construye un "mayordomo" que abría la puerta y saludaba a los visitantes. Ya en la edad moderna (s. XVII), los dos famosos relojeros de centro Europa, construyeron tres androides: un niño que escribía, otro que dibujaba y una joven que tocaba el órgano y simulaba respirar. Esta realización, basada en mecanismos de relojería, les costó ser detenidos y encerrados por la Inquisición. En cuanto a los avances del Siglo XIX y del Siglo XX, hay que destacar los trabajos de Pascal, Leibnitz, Babbage y Boole. También es importante la aportación de Ada Lovelace, colaboradora de Charles Babbage y mujer de Lord Byron, que en el conocido "régimen de Lovelace" establece lo siguiente: "las

máquinas sólo pueden hacer todo aquello que sepamos cómo ordenarles que hagan. Su misión es ayudar a facilitar lo ya conocido". Esto que está todavía vigente en la "informática convencional" fue superado por los avances en IA. También cabe destacar y es revelador el hecho de que varios de los padres indiscutibles de la ciencia de la computación, como son Turing, Wiener y Von Neumann, dedicaran un gran esfuerzo al estudio de la inteligencia humana.

La contribución del matemático inglés Alan Turing al mundo de la computación en general, y al de la IA en particular, fue muy considerable. Turing, participó en el diseño de uno de los primeros computadores que existieron, desarrollado para el ejército inglés, entre los años 40 y 50. Además de aportar los conceptos básicos de la arquitectura secuencial de los computadores, publicó en 1950 un provocador artículo que tiene por título *Computer Machinery and Intelligence*, que comienza con la no menos provocadora pregunta: ¿Pueden las máquinas pensar? En dicho artículo el autor intenta abordar formalmente acciones consideradas hasta ese momento propias de los seres humanos, de forma que después pudieran implementarse en las máquinas, dotándolas así de capacidades como: aprender, crear, etc.

Sin embargo, en la IA tradicional (escuela del MIT), que tuvo mayor auge en los primeros veinte años, los investigadores se encontraron con que sus sistemas sucumbían ante la creciente longitud y complejidad de su programación. Stewart Wilson, investigador del Instituto Roland (Massachussets), se dio cuenta de que algo andaba mal en el campo de la IA "tradicional" y, preguntándose cuáles eran las raíces de la inteligencia, se convenció de que la mejor manera de comprender cómo funciona algo en un ser humano, es comprenderlo primero en un anima1 más simple. Teniendo en cuenta que, en última instancia, la IA intentaba conseguir una réplica de la inteligencia humana, Wilson decidió que lo primero que había que hacer era replicar la inteligencia animal. Se trataba de una idea que nunca había tenido mucha popularidad entre los investigadores de IA, pero él y

otros investigadores pronto la transformaron en un primer principio informal de un nuevo enfoque de ésta, basado en la naturaleza.

2.1.1. Enfoques de la Inteligencia Artificial

Se pueden definir dos puntos de vista o enfoques de la IA, el punto de vista tecnológico o ingenieril y el punto de vista científico.

Por un lado, la rama ingenieril busca la creación de sistemas informáticos que realicen tareas para las que se precisa inteligencia. Se persigue desde este planteamiento la resolución de problemas concretos, sin limitar las técnicas a utilizar a aquellas que utilizan los seres inteligentes.

Por otro lado, la rama científica de la IA se puede definir como "el estudio del comportamiento inteligente, siendo su fin conseguir una teoría de la inteligencia que explique la conducta que se produce en seres de natural inteligentes, y que guíe la creación de entes artificiales capaces de alcanzar dicho proceder inteligente".

Las técnicas clásicas, desde el punto de vista tecnológico han tenido un relativo éxito, y sus productos (Sistemas Expertos, Sistemas Basados en el Conocimiento, etc.) se usan ampliamente. El principal problema de estas técnicas radica en que no son capaces de adaptarse a los cambios del entorno y que es preciso tener un conocimiento explícito del problema para poder abordarlo satisfactoriamente. Estos sistemas han de ser programados y no pueden auto programarse y adaptarse así a nuevos requisitos del entorno. Para resolver este problema, se han desarrollado diversas aproximaciones computacionales conocidas de forma global como Técnicas Adaptativas. A continuación, se describirán las más utilizadas.

2.2. Agentes Inteligentes

Un agente es un ente capaz de percibir su entorno a través de sensores y actuar sobre ese ambiente a través de efectores. Un agente humano tiene ojos, oídos y otros órganos como

sensores; las manos, las piernas, la boca y otras partes del cuerpo como efectores. Un agente robótico sustituye cámaras y telémetros infrarrojos por los sensores y varios motores como efectores. Un agente de software ha codificado cadenas de 27 bits como sus percepciones y acciones. (Un agente genérico se esquematiza en la Figura 2.1).

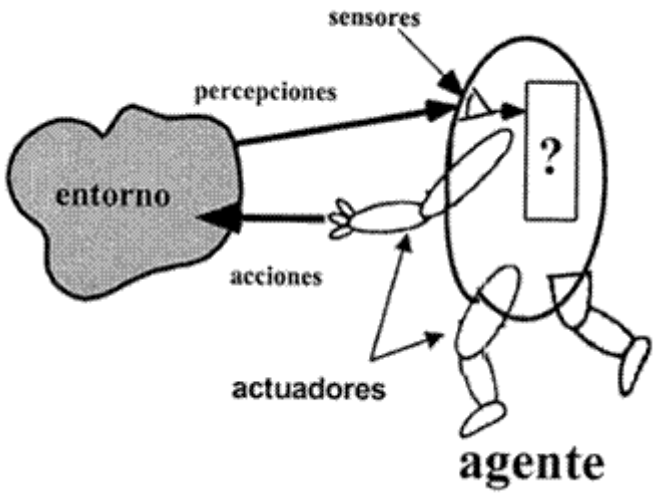


Figura 2.1. Interacción de agentes con su ambiente a través de sensores y efectores.
Fuente: Russel, 2010.

Usamos el término percepción para referirse a los insumos de percepción del agente en un momento dado. La secuencia de percepciones de un agente es la historia completa de todo lo que el agente haya percibido. En general, la elección de un agente de acción en cualquier instante dado puede depender de toda la secuencia de percepciones observado hasta la fecha, pero no en cualquier cosa que no ha percibido. Al especificar la elección del agente de la acción para cada secuencia de percepciones posible, hemos dicho más o menos todo lo que hay que decir sobre el agente. Matemáticamente hablando, se dice que el comportamiento de un agente es descrito por la función de agente que mapea cualquier secuencia de percepciones dado a una acción.

Un autor sostuvo “Se define como agente inteligente a una entidad autónoma y flexible que observa y actúa sobre un entorno con la finalidad de cumplir sus objetivos. Se entiende por

flexible a la capacidad ser reactivo, pro-activo y social. Un agente es reactivo cuando es capaz de responder a los cambios de su entorno, es pro-activo cuando es capaz de intentar cumplir sus propios objetivos y es social cuando es capaz de comunicarse con otros agentes".

2.2.1. Entornos de trabajo de los agentes inteligentes

La gama de entornos de trabajo que pudiera surgir en la IA es obviamente enorme. Podemos, sin embargo, identificar un número bastante reducido de dimensiones en las que los entornos de trabajo pueden ser categorizados. Estas dimensiones determinan, en gran medida, el diseño agente apropiado y la aplicabilidad de cada una de las principales familias de técnicas para la aplicación del agente. A continuación, analizamos varios ambientes de trabajo en los que trabajan los agentes inteligentes.

- **Totalmente observable vs. Parcialmente observable**: Si los sensores de un agente le dan acceso al completo estado del medio ambiente en cada punto en el tiempo, entonces se dice que el ambiente de trabajo es completamente observable. Un entorno de tarea es efectivamente completamente observable si los sensores detectan todos los aspectos que son relevantes para la elección de la acción; relevancia, a su vez, depende de la medida de rendimiento.

  Ambientes totalmente observables son convenientes porque el agente no necesita mantener cualquier estado interno para no perder de vista el mundo. Un ambiente podría ser parcialmente observable debidos sensores ruidosos e inexactas o porque partes del estado están simplemente ausentes en el ejemplo de datos de sensor, un agente de vacío con solamente un sensor de suciedad local no puede decir si hay suciedad en otras plazas, y un taxi automatizado no puede ver lo que otros comensales están pensando. Si el agente no tiene sensores, entonces el ambiente es inobservable.

- **Agente individual vs. Multi-agente:** La distinción entre un solo agente y entornos multi-agente puede parecer bastante simple. Por ejemplo, un agente de resolver un crucigrama por sí mismo es claramente en un entorno de un solo agente, mientras que en un juego de ajedrez se encuentra en un entorno *twoagent* (dos agentes). Hay, sin embargo, algunos problemas sutiles. ¿Un agente de A (el conductor del taxi, por ejemplo) tiene que tratar un objeto B (otro vehículo) como agente. o puede ser tratado simplemente como un objeto que se comporta de acuerdo con las leyes de la física, de forma análoga a las ondas en la playa o las hojas en el viento? La distinción clave es si la conducta de B se describe mejor como la maximización de una medida de desempeño cuyo valor depende de la conducta de un agente.

  Por lo tanto, el ajedrez es un entorno multi-agente competitivo. En el entorno de taxi conducción, por otro lado, evitar colisiones maximiza la medida de rendimiento de todos los agentes, por lo que es un entorno de multi-agente parcialmente cooperativa. También es parcialmente competitiva porque, por ejemplo, sólo un coche puede ocupar una plaza de aparcamiento. Los problemas agente-diseño en entornos multi-agente son a menudo muy diferentes de aquellos en los entornos de un solo agente; por ejemplo, la comunicación a menudo surge como un comportamiento racional en entornos multi-agente; en algunos entornos competitivos, el comportamiento aleatorio es racional porque evita los escollos de la previsibilidad.
- **Determinista vs. Estocástico:** Si el siguiente estado del medio ambiente está completamente determinado por el estado actual y la acción ejecutada por el agente, entonces decimos que el medio ambiente es determinista; de lo contrario, es estocástico. En principio, un agente no necesita preocuparse por la incertidumbre en un entorno completamente observable, determinista. Si el ambiente es parcialmente observable, entonces podría parecer ser estocástica. La mayoría de las situaciones reales son tan complejos que es imposible hacer un seguimiento de todos los aspectos no observados; para fines prácticos, deben ser tratados como estocástico.

Conducción de taxi es claramente estocástico en este sentido, porque uno nunca puede predecir el comportamiento del tráfico exactamente [Russel, 2010]

Decimos que un ambiente es incierto si no es completamente observable o no determinista. Una nota final: el uso de la palabra estocástico generalmente implica que la incertidumbre acerca de los resultados se cuantifica en términos de probabilidades; un entorno no determinista es uno en el que las acciones se caracterizan por sus posibles resultados, pero no hay probabilidades están unidos a 30 ellos. Descripciones de entorno no deterministas se asocian generalmente con las medidas de rendimiento que requieren el agente para tener éxito para todos los posibles resultados de sus acciones.

- **Episódica vs Secuencial:** En una episódica tarea de entorno, la experiencia del agente se divide en episodios atómicos. En cada episodio el agente recibe una percepción y luego realiza una sola acción. Fundamentalmente, el próximo episodio no depende de las acciones llevadas a cabo en los episodios anteriores. Muchas de las tareas de clasificación son episódicos. Por ejemplo, un agente que tiene que detectar piezas defectuosas en unas bases de la línea de montaje de cada decisión sobre la parte actual, independientemente de las decisiones anteriores; por otra parte, la decisión actual no afecta si la siguiente parte es defectuosa. En entornos secuenciales, por otro lado, la decisión actual podría afectar a todas las decisiones futuras. Ajedrez y conducción secuencial de taxis: en ambos casos, las acciones a corto plazo pueden tener consecuencias a largo plazo. Entornos episódicos son mucho más simples que los entornos secuenciales porque el agente no tiene que pensar en el futuro.
- **Estático vs. Dinámico:** Si el entorno puede cambiar mientras que un agente está deliberando, entonces decimos que el medio ambiente es dinámico para el agente; de lo contrario, es estático. Ambientes estáticos son fáciles de tratar debido a que el agente no tiene por qué seguir buscando en el mundo mientras se está decidiendo sobre una acción, ni necesita preocuparse por el paso del tiempo. Los entornos

dinámicos, por otro lado, están pidiendo continuamente lo que el agente quiere hacer; si no se ha decidido todavía, que cuenta como la decisión de no hacer nada. Si el ambiente en sí no cambia con el paso del tiempo, pero las acciones del agente cambian el entorno, entonces decimos que el medio ambiente es semidinamico. Conducción de taxi es claramente dinámico: los otros coches y el propio taxi de mantenerse en movimiento, mientras que el algoritmo de conducción vacila sobre qué hacer a continuación. Ajedrez, cuando se juega con un reloj, es semidinamico. Los crucigramas son estáticos.

- **Discreto vs. Continua:** La distinción discreta / continua se aplica al estado del medio ambiente, a la forma en el tiempo se manipula, ya las percepciones y acciones del agente. Por ejemplo, el entorno de ajedrez tiene un número finito de estados distintos (excluyendo el reloj), el ajedrez también tiene un conjunto discreto de percepciones y acciones. La conducción de taxi es un problema de estado continuo y en tiempo continuo: la velocidad y la ubicación del taxi y de los otros vehículos barren a través de una gama de valores continuos y lo hacen sin problemas el paso del tiempo. Las acciones de conducción de un taxi son también (ángulos de dirección, etc.) continuos. La fotografía de las cámaras digitales es discreta, en sentido estricto, pero se trata típicamente como la representación continua intensidades variables y lugares. [Russel, 2010].

- **Conocido vs. Desconocido:** En sentido estricto, esta distinción no se refiere al medio ambiente en sí, sino al estado de diseñador de conocimiento acerca de las "leyes de la física" del entorno del agente. En un entorno conocido, se dan los resultados (o probabilidades de resultado si el ambiente es estocástico) para todas las acciones. Obviamente, si el ambiente es desconocido, el agente tendrá que aprender cómo funciona con el fin de tomar buenas decisiones. Tenga en cuenta que la distinción entre los entornos conocidos y desconocidos no es la misma que la que existe entre entornos total y parcialmente observables. Es muy posible que un

entorno conocido por ser parcialmente observable, por ejemplo, en los juegos de cartas “solitario”, conozco las reglas, pero todavía soy incapaz de ver las cartas que aún no han sido entregados. Por el contrario, un entorno desconocido puede ser plenamente observables en un nuevo juego de vídeo, la pantalla puede mostrar todo el estado del juego, pero todavía no sé lo que hacen los botones hasta que yo los trato.

2.3. Ingeniería de Software Orientado a Agentes

Las arquitecturas de software que contienen muchos componentes interactuando dinámicamente, cada uno con su propio hilo de control, e involucrándose en protocolos de interacción complejos, son típicamente de órdenes de magnitud más complejas para ingeniarlos correctamente y eficientemente, que aquellos que simplemente computan una función de alguna entrada en un hilo de control.

En muchas aplicaciones de la vida real esta es la característica, por lo cual se han desarrollado herramientas y técnicas para modelar, entender, e implementar sistemas en los que las interacciones son la norma.

Desde los 80s los sistemas de software de agentes y sistemas multi-agentes han crecido en lo que es una de las áreas más activas de investigación y desarrollo en la computación. Una de las razones más importantes es la de los agentes como sistemas autónomos, capaces de interactuar con otros agentes para satisfacer sus objetivos de diseño.

El objetivo de este artículo es revisar el estado del arte en ingeniería de software orientada a agentes. Se da un concepto de lo que son agentes y sistemas multi-agentes, y se comenta la relación entre los agentes y los objetos (desde el punto de vista de OOP).

Luego se revisan algunas metodologías preliminares para ingeniería de sistemas multi-agentes – las cuales proveen una aproximación estructural pero no matemática al análisis y

diseño de los sistemas de agentes, y que toman mucha de su inspiración de las metodologías de análisis y diseño orientado a agentes y aproximaciones de ingeniería de conocimiento. Además, se comentan métodos formales para ingeniería de sistemas multi-agentes.

Por último, se discuten problemas, retos y aspectos que deben ser manejados si se espera que los agentes logren su potencial como un paradigma de ingeniería de software.

2.3.1. Análisis y diseño de agentes

Las metodologías de desarrollo informales para el análisis y desarrollo de sistemas basados en agentes pueden ser ampliamente divididos en dos grupos:

- Las que tienen su inspiración en el desarrollo orientado a objetos, y extienden metodologías orientadas a objetos existentes o las adaptan para los propósitos de la Ingeniería de Software Orientada a Agentes.
- Las que adaptan la ingeniería de conocimiento u otras técnicas.

Como ejemplos de la primera se tienen la AAII Methodology de Kinny et al, Gaia de Wooldrige et al, Agent UML de Odell et al. Como representantes de la segunda se tienen DESIRE de Treur et al, Cassiopeia de Collinot et al y una especificación en el lenguaje Z realizada por Luck and d´Inverno

2.3.2. Métodos Formales para AOSE

Una de las áreas de trabajo más activas en Ingeniería de Software Orientado a Agentes ha sido el uso de métodos formales. Hablando de manera amplia, los métodos formales juegan tres roles en la ingeniería de software:

- Especificación de sistemas.
- Programación de sistemas directamente.
- Verificación de sistemas.

2.4. Agente inteligente

Los agentes nacieron de la investigación en Inteligencia Artificial (IA) y más concretamente de la Inteligencia Artificial Distribuida (IAD), que integra la IA con los Sistemas Distribuidos. Un agente es una entidad física o abstracta que percibe su ambiente a través de sensores, es capaz de evaluar las percepciones y tomar decisiones por medio de mecanismos de razonamiento (Figura 2.2), comunicarse con otros agentes para obtener información y actuar sobre su entorno a través de efectores (Fernández-Caballero, Gascueña, 2009).

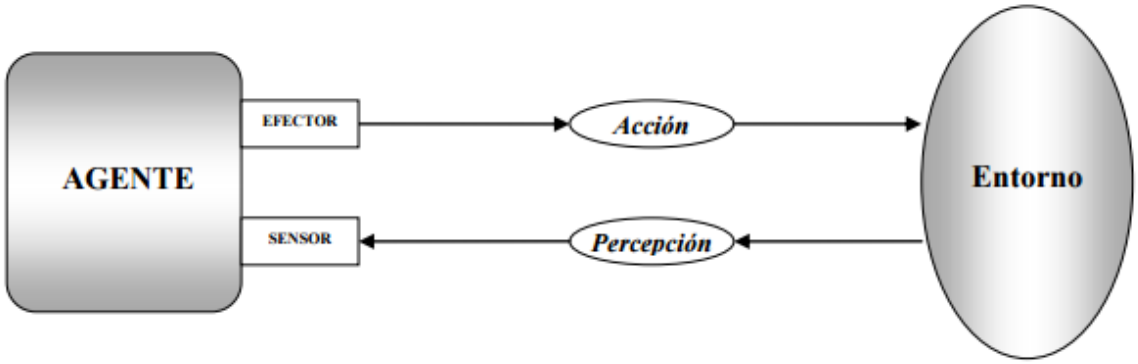


Figura 2.2 Modelo de agente
Fuente: Fernández-Caballero, Gascueña, 2009.

Realizan acciones flexibles para conseguir sus objetivos como:

- Reactivo, el agente es capaz de responder a cambios en el entorno en que se encuentra situado.
- Pro-activo, a su vez e agente debe ser capaz de intentar cumplir sus propios planes u objetivos.

- Social, debe poder comunicarse con otros agentes mediante algún tipo de lenguaje de comunicación de agentes. Interacción con otros agentes como: Cooperación, competencia, negociación.

A grandes rasgos, las características típicas de un agente inteligente son:

- Continuidad Temporal, un proceso sin fin, ejecutándose continuamente y desarrollando su función.
- Autonomía, un agente es completamente autónomo si es capaz de actuar basándose en su experiencia. El agente es capaz de adaptarse, aunque el entorno cambie severamente. Por otra parte, una definición menos estricta de autonomía seria cuando el agente percibe el entorno.
- Sociabilidad, el atributo de la sociabilidad permite a un agente comunicarse con otros agentes o incluso con otras entidades.
- Racionalidad, el agente siempre realiza ¨lo correcto¨ a partir de los datos que percibe del entorno.
- Reactividad, un agente actúa como resultado de cambios en su entorno. En este caso, un agente percibe el entorno y estos cambios dirigen el comportamiento del agente.
- Pro-actividad, un agente es pro-activo cuando es capaz de controlar sus propios objetivos a pesar de cambios en el entorno.
- Adaptabilidad, está relacionado con el aprendizaje que un agente es capaz de realizar y si puede cambiar su comportamiento basándose en ese aprendizaje.
- Movilidad, capacidad de un agente de trasladarse a través de una red telemática.
- Veracidad, asunción de que un agente no comunica información falsa a propósito.
- Benevolencia, asunción de que un agente está dispuesto a ayudar a otros agentes si esto no entra en conflicto con sus propios objetivos.

2.5. Sistemas Multi-Agente

Un agente puede trabajar solo o en cooperación con otros agentes con el fin de satisfacer uno o varios objetivos. En el caso de cooperación se habla de Sistemas multi-Agente (SMA), es decir, sistemas formados por un conjunto de agentes que interaccionan entre sí para alcanzar así objetivos comunes. Estos sistemas se sitúan en un entorno que proporciona a cada agente información que modifica su estado mental, activando mecanismos de razonamiento que desencadenan la ejecución de acciones y la comunicación con otros agentes para alcanzar los objetivos marcados. Integrados entorno, debe haber objetos de tal forma que puedan ser percibidos, creados, destru modificados por los agentes. Por esto, cada agente debe disponer de un conjunto de operaciones que le permitan percibir, producir, consumir, transformar y manipular dichos objetos. Además, cada agente debe perseguir uno o varios objetivos. Por lo tanto, para poder hablar de un SMA, es necesario que el sistema reúna los siguientes elementos: un conjunto de agentes, un entorno y un conjunto de objetos.

Existen varias metodologías para desarrollar estos sistemas y cada una de ellas tiene unas características particulares que la hacen más adecuada para ciertos proyectos.

A continuación, se describen brevemente algunas de las metodologías más utilizadas en el desarrollo de SMA.

- MAS-CommonKADS. Extiende al campo de los SMA la metodología CommonKADS, para sistemas expertos, utilizando estructuración orientada a objetos y lenguajes de especificación de protocolos. Gira alrededor del modelo de experiencia y está pensada para desarrollar sistemas expertos que interactúen con el usuario. Además, plantea el desarrollo de SMA integrado en el ciclo de vida software denominado espiral.
- MaSE (*Multi-agent systems Software Engineering*). Parte del paradigma orientado a objetos y asume que un agente es la especialización de un objeto. Los agentes se

coordinan unos con otros vía conversaciones y actúan proactivamente para alcanzar metas individuales y globales. El proceso de desarrollo en MaSE consta de las siguientes etapas: captura de objetivos, aplicación de casos de uso, refinamiento de roles, creación de las clases de agentes, construcción de las conversaciones, ensamblaje de las clases de agentes y diseño del sistema. La mayoría de estas fases están soportados por la herramienta *AgentTool*.

- ZEUS. Combina distintos resultados de investigación en agentes (planificación, ontologías, asignación de responsabilidades, relaciones sociales entre agentes) en un sistema ejecutable. ZEUS propone un desarrollo en cuatro etapas: el análisis del dominio, el diseño de los agentes, la realización de los agentes y el soporte en tiempo de ejecución. Estas etapas están basadas en el uso de roles para analizar el dominio y la asignación de roles a agentes.
- GAIA. Propone cómo realizar un análisis basado en los roles del SMA. El objetivo es comprender el sistema y su estructura, sin referenciar ninguna característica de la implementación, con el fin de obtener un diseño suficientemente detallado como para poder ser implementado directamente. Esto se consigue a través de la idea de organización, la cual es una colección de roles que mantienen relaciones entre sí y toman parte en patrones institucionalizados de interacción con otros roles. Los roles agrupan cuatro aspectos: las responsabilidades del agente, los recursos que se le permite utilizar, las tareas asociadas y las interacciones.

Las metodologías Prometheus e INGENIAS son de especial interés utilizada en el desarrollo de este proyecto. En las siguientes secciones se describen detalladamente ambas metodologías.

2.5.1. Metodología PROMETHEUS

La metodología Prometheus define un proceso detallado para especificar, diseñar, implementar y probar/depurar sistemas software orientados a agentes. Propone un proceso

de desarrollo iterativo que consta de tres fases: especificación del sistema, diseño arquitectónico y diseño detallado. Además, detalla cómo seguir el proceso y qué resultados se obtienen en cada fase (Padgham, Winikoff, 2004).

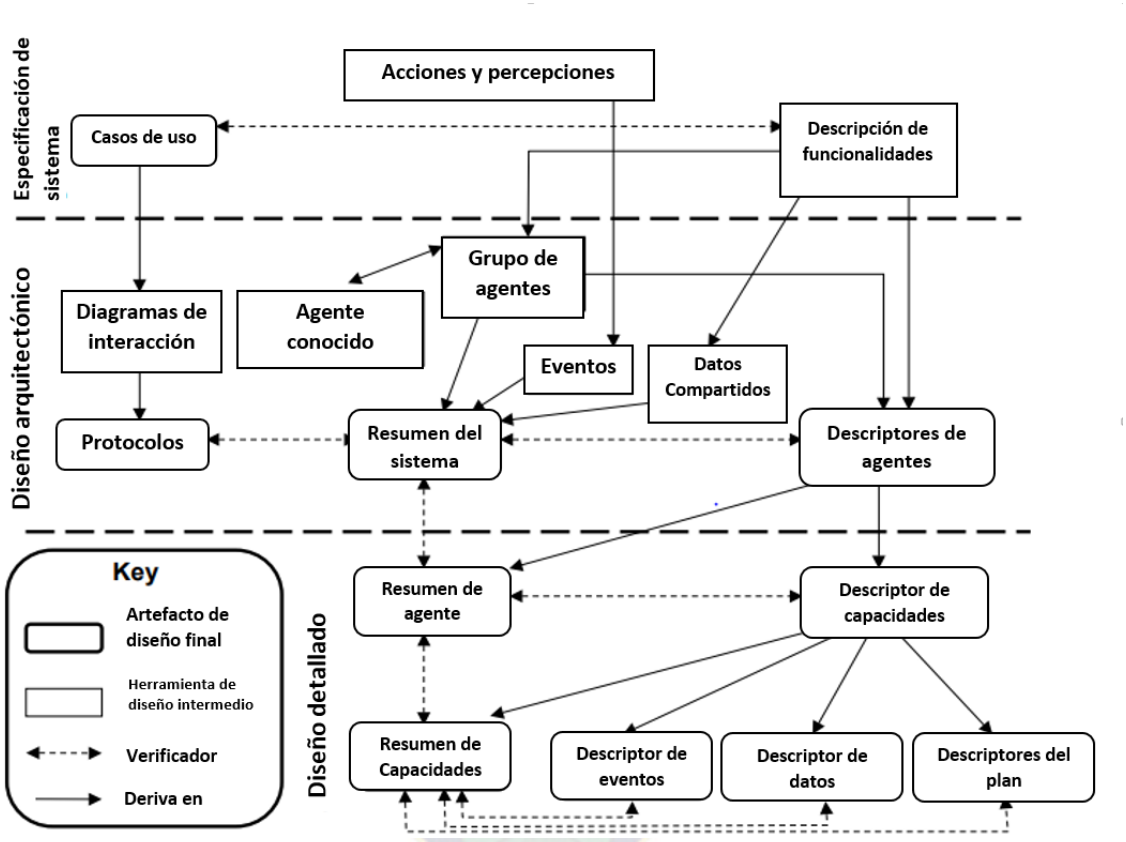


Figura 2.3 Diagrama metodología Prometheus
Fuente: Padgham, Winikoff, 2004

2.5.1.1. Fase de especificación del sistema

Consiste en identificar los objetivos y funcionalidades básicas del sistema, desarrollar los escenarios que ilustran el funcionamiento del sistema y especificar las entradas y salidas del sistema (percepciones y acciones, respectivamente). Los objetivos son un mecanismo para especificar los requisitos del sistema, es decir, permiten identificar por qué razones se está construyendo el sistema. El proceso de especificación de objetivos se recomienda hacer en

dos pasos. El primero consiste en identificar los objetivos iníciales a partir de los requisitos proporcionados. El segundo paso consiste en refinar los objetivos agrupándolos por sus similitudes, de tal modo que pueda obtenerse un grupo de objetivos por cada funcionalidad que deba soportar el sistema. Es preciso resaltar que una funcionalidad describe un fragmento del comportamiento del sistema. Los escenarios describen el funcionamiento del sistema ilustrando la ejecución normal del sistema, así como las situaciones en las que se den sucesos anormales (alternativos). Un escenario se especifica en varios pasos, haciendo evidente dónde se necesita información del entorno y dónde es necesario realizar actuaciones. La información recogida del entorno se denomina percepción en la metodología Prometheus y constituye la entrada al sistema.

2.5.1.2. Fase de diseño arquitectónico

Toma los artefactos producidos en la fase anterior para identificar los tipos de agentes que existirán en el sistema y cómo interactuarán entre ellos para alcanzar los objetivos del sistema. Cada agente debe ser responsable de al menos una de las funcionalidades identificadas en la fase anterior. Por ello, para identificarlos se agrupan funcionalidades considerando todas las alternativas posibles de tal modo que finalmente se considere aquella agrupación que tenga una mayor cohesión y un mínimo acoplamiento. Otra tarea importante en esta fase consiste en especificar las interacciones que se producen entre agentes. En este caso, estas se describen mediante protocolos de interacción que incluyen los mensajes que se envían los agentes entre sí, qué percepciones llegan a los agentes y qué acciones efectúan sobre el entorno.

2.5.1.3. Fase de diseño detallado

Consiste en desarrollar la estructura interna de cada agente identificado previamente y detallar cómo llevar a cabo sus tareas dentro del sistema global. Para ello, se especifican las capacidades necesarias que cada agente tiene para satisfacer sus responsabilidades, además

de desarrollar los protocolos con el objetivo de indicar el procesamiento interno de cada agente individualmente.

Una vez especificado el sistema se utiliza la herramienta Prometheus Design Tool (PDT) (Padgham, Thangarajah, Paul) para generar el código que sirve de base para implementar la aplicación en el lenguaje de programación Jack.

La herramienta PDT permite llevar a cabo el diseño de un SMA siguiendo la metodología Prometheus. Proporciona una ventana principal dividida en cuatro áreas: la lista de diagramas, la vista del diagrama, la lista de entidades y la vista de descripción de entidades.

- La lista de diagramas presenta de manera ordenada cada una de las fases de la metodología (especificación del sistema, diseño arquitectónico y diseño detallado) e incluye los diferentes tipos de diagramas que se crean en cada una de ellas.
- La vista del diagrama permite visualizar y especificar el diagrama seleccionado en la lista de diagramas. Para ello ofrece una paleta con las entidades que pueden incluirse en el diagrama seleccionado.
- La lista de entidades muestra de forma ordenada todas y cada una de las entidades existentes en el diseño del sistema, es decir, muestra todos los agentes, los actores, las percepciones, las acciones, los mensajes, los roles, los datos, los escenarios y los protocolos de interacción que forman parte de cada uno de los diagramas que componen el diseño del sistema.
- La vista de descripción de entidades muestra una descripción detallada de la entidad seleccionada en la lista de entidades o en la vista del diagrama. Además, permite añadir o modificar cualquier detalle de la descripción.

2.5.1.4. Prometheus Design Tool

El menú de la herramienta ofrece, a parte de las opciones esperadas en una herramienta de este tipo, una opción denominada Crosscheck para comprobar la consistencia del diseño y

conocer, antes de generar el código, si el diseño obtenido se ajusta por completo a lo especificado en la metodología Prometheus. Otro detalle relevante es la posibilidad de utilizar la opción Complete HTML Report para generar automáticamente un informe completo de la especificación en HTML. Por último, comentar que, tras completar el diseño, comprobar la consistencia y generar el informe.

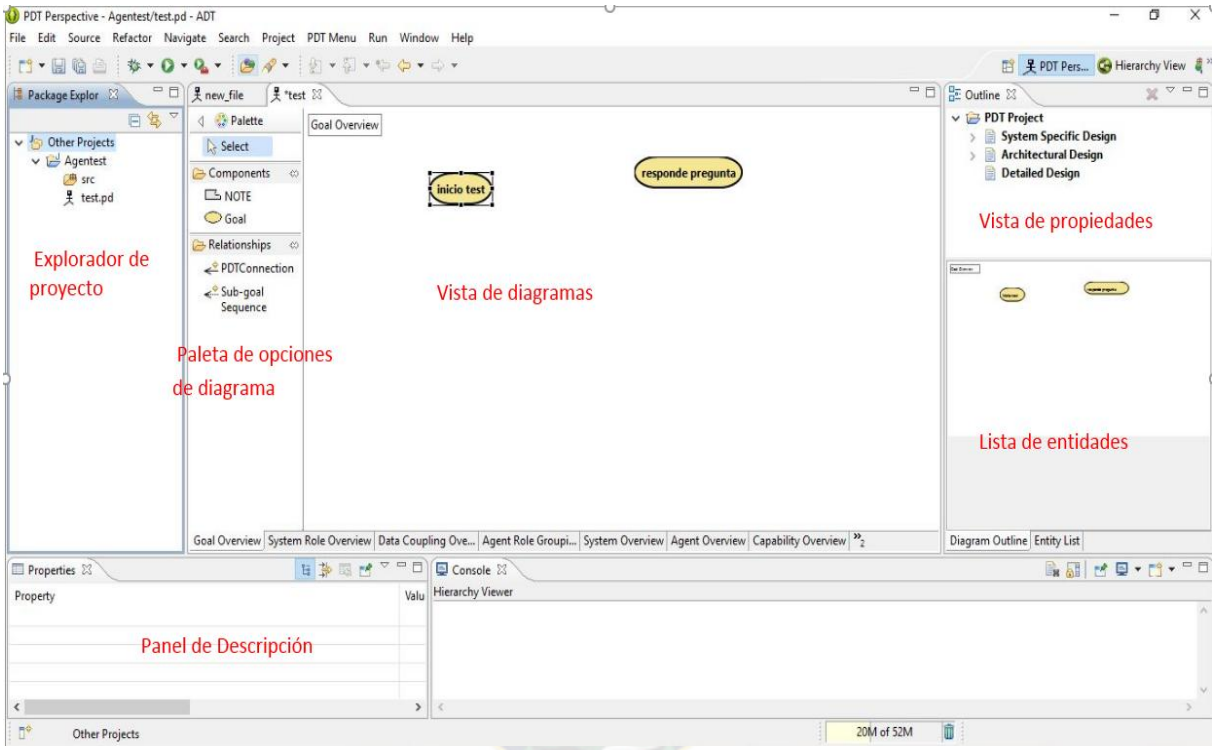


Figura 2.4 Descripción de entorno de herramienta Prometheus

A continuación, se describen algunos conceptos fundamentales utilizados en la metodología Prometheus:

- **Percepción.** Ítem de información del entorno recogida por algún tipo de sensor.
- **Acción.** Aquello que hace el agente y que puede ser de dos tipos, acción externa y acción interna. Una acción externa es la que produce cambios en el entorno, mientras que una acción interna es aquella que no afecta al entorno y que necesita realizar el agente, como, por ejemplo, el acceso a una base de datos.

- **Objetivo.** Algo en lo que está trabajando el agente y hacia lo que se dirige. - Evento. Un suceso significativo al que el agente debe responder de alguna forma. Una percepción es un tipo particular de evento.
- **Mensaje.** Unidad de interacción entre agentes, la cual se utiliza siguiendo un protocolo definido.
- **Protocolo.** Definición de patrones de interacción permitidos.

Figura 2.5. Simbología usada en (PDT).
Fuente: Padgham, Winikoff, 2004

En la figura 2.5 se muestra simbología utilizada para los distintos esquemas en la herramienta Prometheus Design Tool.

2.5.2. Metodología INGENIAS

La metodología INGENIAS (Gómez, Pavón) propone un proceso de desarrollo basado en el Proceso Unificado de Desarrollo de Software (*Unified Software Development Process o*

RUP) para desarrollar SMA Durante este proceso se generan una serie de modelos que se detallan a continuación:

- **Modelo de organización.** Describe la estructura del SMA, los roles y los flujos de trabajo (*workflows*).
- **Modelo de entorno.** Describe las entidades y relaciones con el entorno del SMA.
- **Modelo de agente**. Describe los agentes del sistema, así como sus responsabilidades, control y estado mental.
- **Modelo de objetivos y tareas**. Identifica los objetivos y tareas generales del sistema, los cuales pueden asignarse a agentes.
- **Modelo de interacción**. Describe las interacciones existentes entre agentes.

Estos modelos se especifican en los flujos de análisis y diseño (Gómez, 2002), los cuales se dividen a su vez en tres fases, inicio, elaboración y construcción, de acuerdo al modelo iterativo RUP. La fase de análisis-inicio identifica los componentes que utilizan los agentes para interactuar con el entorno y especifica los casos de uso más relevantes. Los componentes se traducen en aplicaciones que especifican las acciones que pueden ejecutar los agentes, así como las percepciones de los agentes. La fase de diseño-inicio consiste en elaborar un prototipo del sistema que muestra cómo interactúa el usuario con la aplicación y qué tipo de resultados se esperan. En la fase de análisis-elaboración se analizan los casos de uso, obtenidos en la etapa anterior, para generar la arquitectura del sistema. Para ello se refinan los casos de uso y se asocian con modelos de interacciones. La fase de diseño-elaboración aumenta el nivel de detalle de la especificación determinando cómo se llevan a cabo los casos de uso identificados. Los resultados que se obtienen se centran en refinar los flujos de trabajo y las tareas asociadas, las interacciones, cómo es el control del agente y cómo se satisfacen los objetivos del sistema. El modelo de organización se refina asociando tareas a los flujos de trabajo. El modelo de interacción se detalla añadiendo relaciones con los flujos de trabajo.

Tras desarrollar los casos de uso necesarios para determinar la arquitectura del sistema, se pasa a la fase de análisis-construcción para identificar los casos de uso restantes sin modificar sustancialmente la visión del sistema que se tiene hasta ahora del sistema.

En la fase de diseño-construcción se detallan las relaciones sociales, principalmente las relaciones de subordinación y las relaciones cliente-servidor.

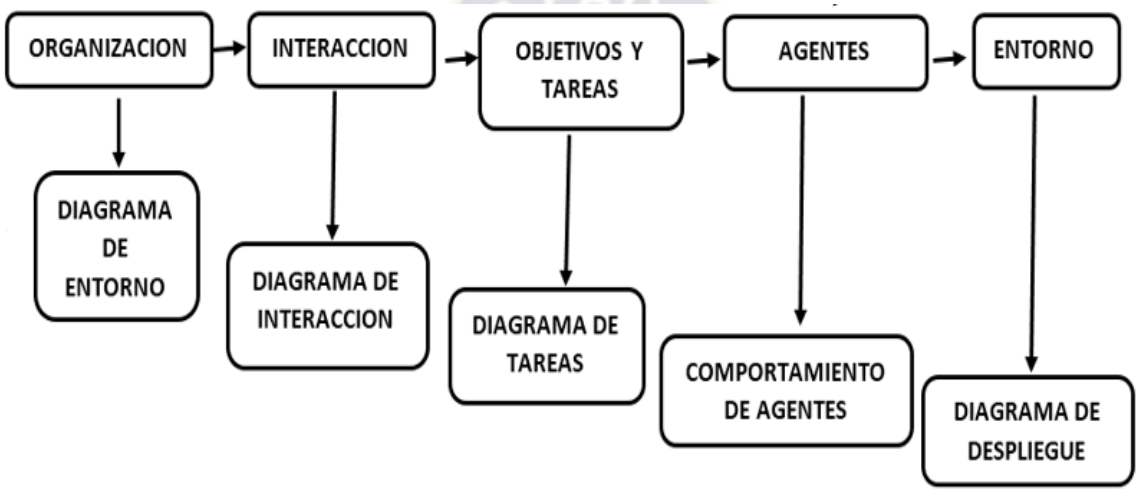


Figura 2.6 diagrama de fases INGENIAS
Fuente: Pavón, J., Gómez-Sanz, J. 2006.

2.5.2.1. INGENIAS Development Kit (IDK)

Es una colección de herramientas para especificar, verificar e implementar SMA. La primera de estas herramientas es un editor gráfico, que permite al desarrollador crear y modificar especificaciones de un SMA utilizando conceptos del lenguaje de INGENIAS. Cada entidad puede visualizarse gráficamente con la notación de UML o utilizando la notación propia definida en la metodología INGENIAS. La ventana principal del editor se divide en cuatro áreas de forma muy similar a la herramienta PDT descrita previamente.

- El área de proyecto muestra en una estructura arbórea todos los diagramas del proyecto. Estos pueden ser de los tipos siguientes: modelo de entorno, modelo de organización, modelo de tareas y objetivos, modelo de interacción, modelo de agente, diagrama de componentes, diagrama de actividad, diagrama de casos de uso, diagrama de interacción AUML y diagrama de despliegue.
- En el área de diagramas se visualizan los diagramas seleccionados en el área del proyecto. En el área de diagramas se crea una pestaña por cada modelo visualizado y ofrece una paleta para crear las entidades específicas de cada tipo de diagrama.
- El área de logs, localizada en la parte inferior del área de diagramas, se utiliza para mostrar informes de la herramienta y para mostrar los detalles descriptivos de la entidad seleccionada.
- El área de entidades muestra jerárquicamente todas las entidades existentes en el modelo.

El segundo tipo de herramientas disponibles en el IDK son los módulos o *pluggins*. Estos requieren de especificaciones de SMA modelados con el lenguaje INGENIAS para verificar sus propiedades y generar automáticamente el código asociado al modelo. El soporte de este tipo de herramientas permite a los desarrolladores extender la funcionalidad de IDK creando sus propios módulos de acuerdo a sus necesidades.

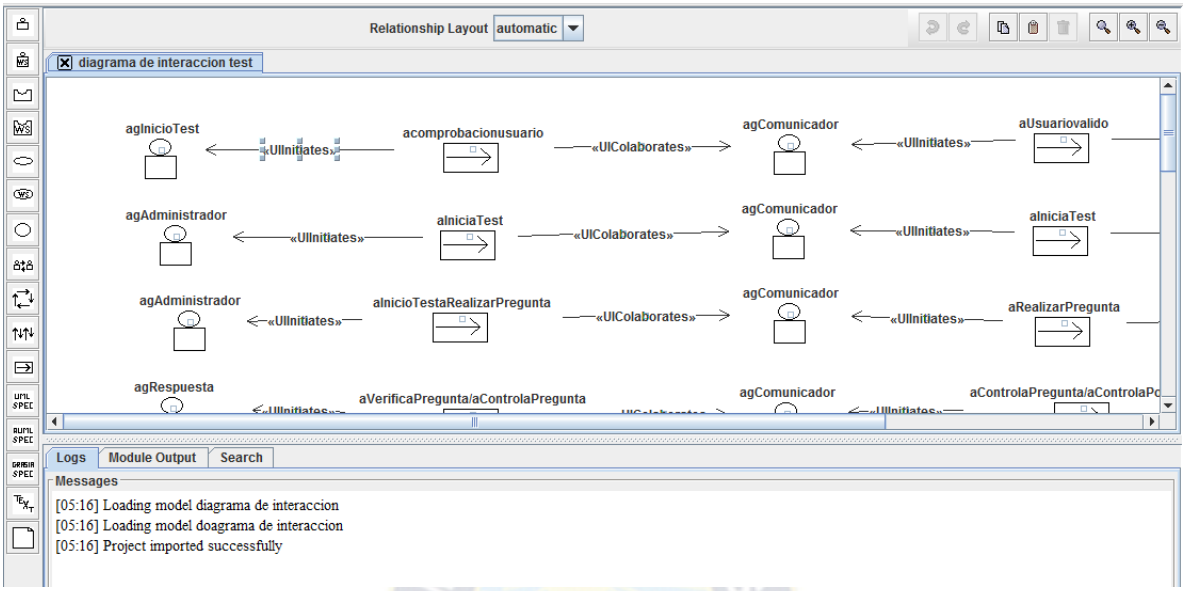


Figura 2.7 entorno INGENIAS DEVELOPMENT KIT (IDK)
Fuente: Fuente: Pavón, J., Gómez-Sanz, J. 2006.

2.6. Test

“Test” es una palabra inglesa que significa “prueba” y que se deriva del latín “testis".

Esta misma raíz figura en palabras españolas tales como testimonio o testigo. La palabra “test” se utiliza sin traducir en todos los países del mundo y sirve para designar una modalidad de exploraciones muy extendida hoy en día en diversos campos científicos y técnicos entre ellos el psicológico (Cerdá, 1978).

Los procedimientos empleados en la elaboración de una prueba también varían con el tipo de esta y los propósitos de los usuarios. Preparar un inventario de lápiz y papel, de intereses o de características de la personalidad, implica problemas diferentes a los de construir una prueba de aprovechamiento o de aptitud. De igual modo, los complejos procedimientos seguidos por los diseñadores profesionales de pruebas son poco familiares para la mayoría de los maestros. Cualquiera que sea el tipo de prueba o las metas de los usuarios, se necesita cierto grado de planeación del contenido antes de escribir los reactivos que contendrá el instrumento. La planeación de la prueba deberá incluir definiciones claras de

las examinadas, las condiciones bajo las cuales se administrará la prueba, e información concerniente a la calificación, interpretación de las puntuaciones y uso que se dará a los resultados.

Existen tipos de test que a continuación se describen:

- Pruebas de observación. Se usa en las pruebas para observar a solicitantes de un trabajo particularmente comienza con un análisis detallado de las actividades a desarrollar en el mismo.
- Pruebas de inteligencia. Son procesos aplicado por diseñadores de pruebas de inteligencia que reúnen un conjunto de reactivos que miden algún aspecto del constructor "inteligencia" estos reactivos son construidos de acuerdo a teorías específicas.
- Inventarios de escala de personalidad. Algunos de estos están basados en el sentido común, y otros en teorías de personalidad y algunos otros en procedimientos estadísticos. Muchos de estos instrumentos son resultado de la combinación de enfoques teóricos, racionales, y empíricos.
- Pruebas de rendimiento. Los procedimientos para la elaboración de pruebas de rendimiento no se limitan a la sola evaluación y motivación del estudiante, se deben dedicar tiempo suficiente para realizar estas pruebas juntamente con un especialista en el tema a evaluar.

Si bien existen variados tipos de pruebas se vio conveniente para el siguiente trabajo el tipo de test con reactivos de opción múltiple que a continuación se describe, este tipo de test de eligió por su facilidad de manejo con el agente.

3.6.1. Reactivos de opción múltiple

No se sabe quién elaboro el primer reactivo de opción múltiple para una prueba, pero desde el punto de vista de la evaluación psicológica fue algo fortuito. Los reactivos de opción

múltiple son los más versátiles de todos los reactivos objetivos, ya que pueden usarse para medir logros de aprendizaje simples y complejos en todos los niveles y en todas las áreas temáticas. Aunque los reactivos de respuesta de ensayo demandan mayor habilidad de organización que la selección de respuestas a los reactivos de opción múltiple, responder de manera correcta a un reactivo de selección múltiple bien preparado requiere una gran habilidad para discriminar y no solo la capacidad para reconocer o recordar la respuesta correcta. Las calificaciones en los reactivos de opción múltiple también son menos afectadas por la adivinación y por otros grupos de respuesta que las calificaciones en otros reactivos objetivos. Además, puede obtenerse información de diagnóstico útil a partir de un análisis de las opciones incorrectas (distractores) seleccionada por los examinados.

Entre los defectos de los reactivos de opción múltiple están que:

1) Los buenos son difíciles de elaborar, en especial aquellos en los que todas las opciones resulten igualmente atractivas para los examinados que no conocen la respuesta correcta.
2) enfatizan el reconocimiento más que el recuerdo y la organización de la información.
3) requieren más tiempo para la respuesta y pueden muestrear el dominio temático de manera menos adecuada que los reactivos de verdadero y falso.

También se ha argumentado, pero no demostrado, que las pruebas de opción múltiple favorecen a los lectores sagaces, hábiles y rápidos, y penalizan a las personas más reflexivas y que piensan con más profundidad (Hoffman, 1962).

En la figura 2.8 se presentan lineamientos para facilitar la elaboración de reactivos de opción múltiple de alta calidad. Tales lineamientos son sobre todo producto de la lógica y de la experiencia, más que de la investigación, y su seguimiento no garantiza la elaboración de buenas pruebas de opción múltiple. La elaboración de buenos reactivos depende mucho o más que del conocimiento de la materia de la prueba, de la comprensión de lo que los

estudiantes deberían saber y de lo que es poco probable que sepan acerca de la materia, y del arte o habilidad de plantear preguntas. Incluso cuando los lineamientos no se siguen con precisión, los reactivos de opción múltiple tienden a ser bastante sólidos en su capacidad para medir el conocimiento y la comprensión.

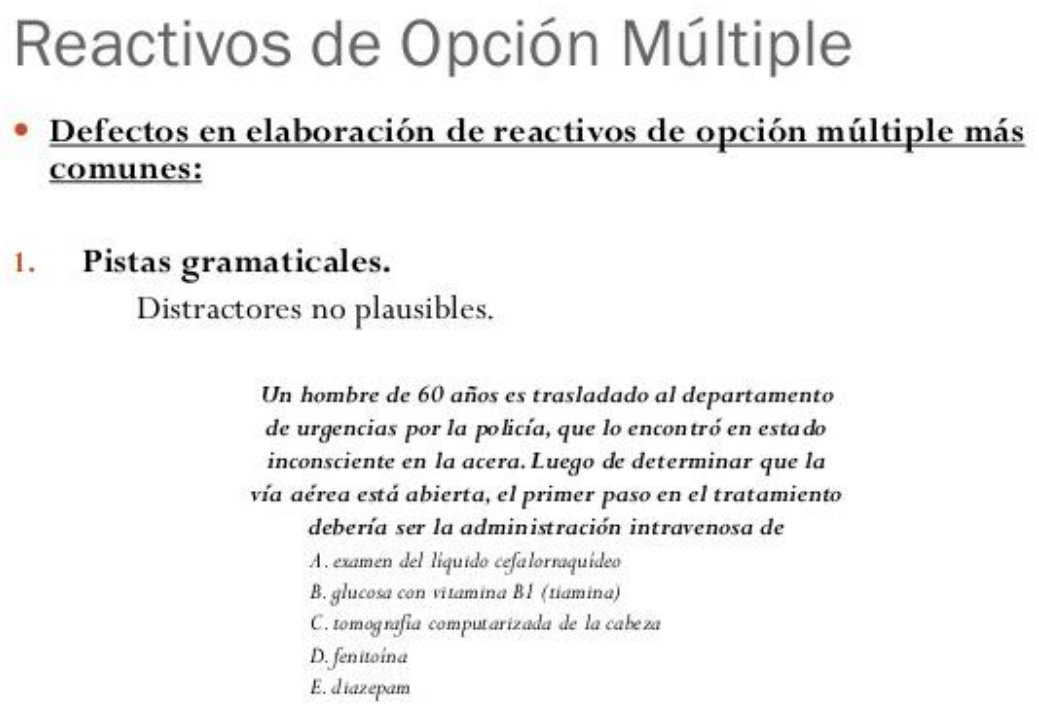


Figura 2.8 Reactivos de opción multiple
Fuente: Case S., Swanson D. NBME 2006

2.7. Propuesta Curricular de ciencias de la computación 2016

La ACM conjuntamente con la IEEE-*Computer Society* teniendo con anticipación una amplia participación a lo largo del tiempo dando directrices curriculares internacionales para los programas de pre-grado en computación, haciendo una retrospectiva de los trabajos con respecto a estas acciones que comenzó con la publicación de una propuesta curricular en el año 1968 hace más de 48 años, se tiene amplios referentes del volumen publicado el

2001 específicamente para Ciencias de la Computación, y un esfuerzo interino de ambas organizaciones concluido el año 2016.

Este volumen, Ciclo de Ciencias de la Computación de 2013 presenta una revisión integral de directrices que incluyen un conjunto de conocimientos redefinidos, el resultado un conjunte de conocimientos esenciales para un programa de ciencias de la computación, también busca ejemplificar cursos y programas adaptados al entorno real para así proporcionar una orientación concreta sobre la estructura de desarrollo en una variedad de contextos institucionales.

Dada la rápida evolución de y expansión del campo. La creciente diversidad de temas potencialmente pertinentes para la educación en ciencias de la Computación y la creciente integración de otras disciplinas crean estos esfuerzos para este propósito, producto de un esfuerzo considerable para involucrar a la comunidad más amplia de la educación en computación y comprender mejor las nuevas oportunidades y necesidades locales aplicadas al modelo curricular.

Este documento se usará como referencia para la base de preguntas del test puesto que es el mejor referente para poder parametrizar los conocimientos y habilidades que un estudiante debe tener, el cual contempla las siguientes áreas:

- AL - Algoritmos y Complejidad
- AR - Arquitectura y Organización
- CN - Ciencias Computacionales
- DS - Estructuras discretas
- GV - Gráficos y visualización
- HCI - Interacción Hombre-Computador
- IAS - Garantía y seguridad de la información
- IM - Gestión de la información

- IS - Sistemas Inteligentes
- NC - Redes y comunicaciones
- SO - Sistemas operativos
- PBD - Desarrollo basado en la plataforma
- PD - Computación paralela y distribuida
- PL - Lenguajes de programación
- SDF - Fundamentos de Desarrollo de Software
- SE - Ingeniería de Software
- SF - Fundamentos de Sistemas
- SP - Cuestiones sociales y práctica profesional.

Las áreas propuestas en el documento Ciclo de Ciencias de la Computación de 2016 presentan áreas que se consideran esenciales para un modelo curricular que hoy en día es de interés en múltiples universidades a nivel mundial.

2.7.1. Redes y Comunicación (NC)

El Internet y las redes de ordenadores son ahora omnipresentes y un número creciente depende fuertemente del correcto funcionamiento de la red subyacente. Las redes, tanto fijos y móviles, son una parte clave del entorno informático de hoy y de mañana. Muchas de las aplicaciones informáticas que se utilizan hoy en día no serían posibles sin las redes.

Esta dependencia de la red subyacente es probable que aumente en el futuro, el objetivo de aprendizaje de alto nivel de este módulo se puede resumir de la siguiente manera:

- Pensar en un mundo en red. El mundo está cada vez más interconectado y el uso de las redes seguirá aumentando. los estudiantes deben entender cómo las redes y los principios clave detrás de la organización y funcionamiento de las redes.

- Continuidad de estudio. El dominio de red está evolucionando rápidamente y una primera red curso debe ser un punto de partida para otros cursos más avanzados sobre diseño de redes, gestión de redes, redes de sensores, etc.
- Los principios y la práctica interactúan. La creación de redes es real y muchas de las opciones de diseño indican que las redes también dependen de limitaciones prácticas. los estudiantes deben estar expuestos a estas limitaciones prácticas experimentando con la creación de redes, el uso de herramientas y la escritura software de red.

"Hay diferentes maneras de organizar un curso de redes. Algunos educadores prefieren un enfoque, es decir, el curso comienza desde las aplicaciones y luego explica la entrega fiable, el enrutamiento y reenvío. Otros educadores prefieren un enfoque de abajo arriba donde los estudiantes empiezan con las capas inferiores y construir su comprensión de la red, transporte y capas de aplicación más adelante". (CS2013).

Se describen los contenidos a continuación:

Introducción (NC)

Temas:

- Organización de Internet (proveedores de servicios de Internet, proveedores de contenidos, etc.)
- Técnicas de conmutación (por ejemplo, circuito, paquete)
- Piezas físicas de una red, incluyendo hosts, routers, switches, ISPs, inalámbricos, LAN, puntos de acceso y cortafuegos
- Principios de estratificación (encapsulación, multiplexación)
- Funciones de las diferentes capas (aplicación, transporte, red, enlace, físico)

Aplicaciones de red (NC)

Temas:

- Sistemas de nombres y direcciones (DNS, direcciones IP, identificadores uniformes de recursos, etc.)
- Aplicaciones distribuidas (cliente / servidor, peer-to-peer, cloud, etc.)
- HTTP como protocolo de capa de aplicación
- Multiplexación con TCP y UDP
- API de socket

Entrega confiable de Datos (NC)

Esta unidad de conocimiento está relacionada con los Fundamentos de Sistemas (SF). Referencia SF / Estado y Maquinas SF / Confiabilidad por Redundancia.

Temas:

- Control de errores (técnicas de retransmisión, temporizadores)
- Control de flujo (confirmaciones, ventana deslizante)
- Problemas de rendimiento (pipelining)
- TCP

Enrutamiento y reenvío (NC)

Temas:

- Enrutamiento versus reenvío
- Enrutamiento estático
- Protocolo de Internet (IP)
- Problemas de escalabilidad (direccionamiento jerárquico)

Redes de área local (NC)

Temas:

- Problema de acceso múltiple
- Enfoques comunes de acceso múltiple (backoff exponencial, multiplexación por división de tiempo, etc.)
- Redes de área local
- Ethernet

- Conmutación

Asignación de recursos (NC)

Temas:

• Necesidad de asignación de recursos

• Asignación fija (TDM, FDM, WDM) versus asignación dinámica

• Enfoques de extremo a extremo versus enfoques asistidos por red

Equidad

• Principios de control de la congestión

• Enfoques de la congestión (por ejemplo, redes de distribución de contenido)

Movilidad (NC)

Temas:

- Principios de las redes celulares
- Redes 802.11
- Problemas en el soporte de nodos móviles (agentes domésticos)

Redes sociales (NC)

Temas:

- Visión general de redes sociales
- Ejemplos de plataformas de redes sociales
- Estructura de los gráficos de redes sociales
- Análisis de redes sociales

Es de los contenidos que se obtiene los parámetros que se usaran en el test, cabe resaltar que los contenidos son quedan cómo una sugerencia para su aplicación al pensum de la carrera de Informática de la UMSA.

CAPÍTULO III

MARCO APLICATIVO

3.1. Introducción

En el presente capítulo se detalla el modelado del sistema multi-agente que administrara el test y los reactivos (preguntas) que alimentaran el mismo, esta combinación ayudara a obtener una probabilidad alta de éxito en la búsqueda por cualificar los conocimientos en los estudiantes de la carrera de informática. Cabe recalcar que para poder posicionar de manera ordenada los reactivos y facilitar a los agentes lograr sus objetivos se pensó en una base multidimensional para poder depositar las preguntas no solo del área de evaluación del presente trabajo como ser **telemática**, si no tener la capacidad de abarcar todas las áreas del campo de la informática.

Se propone en primer lugar seguir las especificaciones del sistema y las fases de diseño arquitectónico propuestas en Prometheus. Así, un modelo inicial de acuerdo con Prometheus se obtiene usando las directrices que permitan identificar los agentes y sus interacciones. Después, se obtendrá un modelo equivalente en INGENIAS utilizando la colección de equivalencias (Mapeos) entre los conceptos utilizados en ambas metodologías. Seguidamente, el modelado continúa con INGENIAS, beneficiándose así de las ventajas de desarrollo de software impulsado por modelos.

Para el desarrollo del modelado del sistema multi-agente se propone realizar una serie de pasos los cuales ya contemplan la unificación de ambas metodologías para poder alcanzar los objetivos propuestos.

A continuación se detalla brevemente los pasos a seguir para obtener el modelado del sistema multi-agente este se encuentra desarrollado en 2 fases:

- **1º Fase: Metodología Prometheus.**

1. **Especificación del sistema.** En esta parte se identificarán los actores que formarán parte del sistema, se desarrollarán los diferentes escenarios del sistema, se definirán las metas de los agentes, y se describirán los roles de los agentes con la base de preguntas.
2. **Arquitectura del sistema.** En esta parte se determinará los tipos de agentes que intervendrán en el sistema tomando como base los roles del sistema y se realizará un análisis descriptivo general que permitirá apreciar cómo será el sistema necesario para poder mejorar el mismo.
3. **Diseño detallado del sistema.** En esta parte se realizará el diagrama detallado del sistema y se realizará la transición y unificación con la metodología INGENIAS.

- **2º FASE Metodología INGENIAS.**

4. **Organización.** En base a lo obtenido a los datos por la transición a INGENIAS se desarrollará el diagrama de entorno el cual reflejará las aplicaciones internas y los agentes que intervienen en estas los cuales formaran el sistema multi-agente.
5. **Interacción.** Se desarrollará el diagrama de interacción de los agentes mostrando un esquema de la comunicación entre ellos por medio de mensajes.
6. **Objetivos y tareas.** se desarrollará el diagrama de tareas a realizar por los agentes, identificando cada una de estas mediante la invocación de métodos realizados previamente en el paso 4.
7. **Agentes.** En esta parte según a lo obtenido en la metodología Prometheus se realizará la especificación de agentes y el diagrama de componentes, los cuales mostraran los estados de los agentes en determinados momentos de cada evento.
8. **Despliegue.** En esta última parte se realizará el diagrama de despliegue donde se mostrará la cantidad de instancias que se generaran para cada entidad.

En la figura 3.1 se muestra el procedimiento de los pasos propuestos estos están plasmados en un diagrama el cual refleja cada una de sus partes y como las partes finales desarrolladas

bajo la metodología Prometheus afecta en la metodología INGENIAS, para de esta manera potenciar el análisis del modelado del sistema multi-agente.

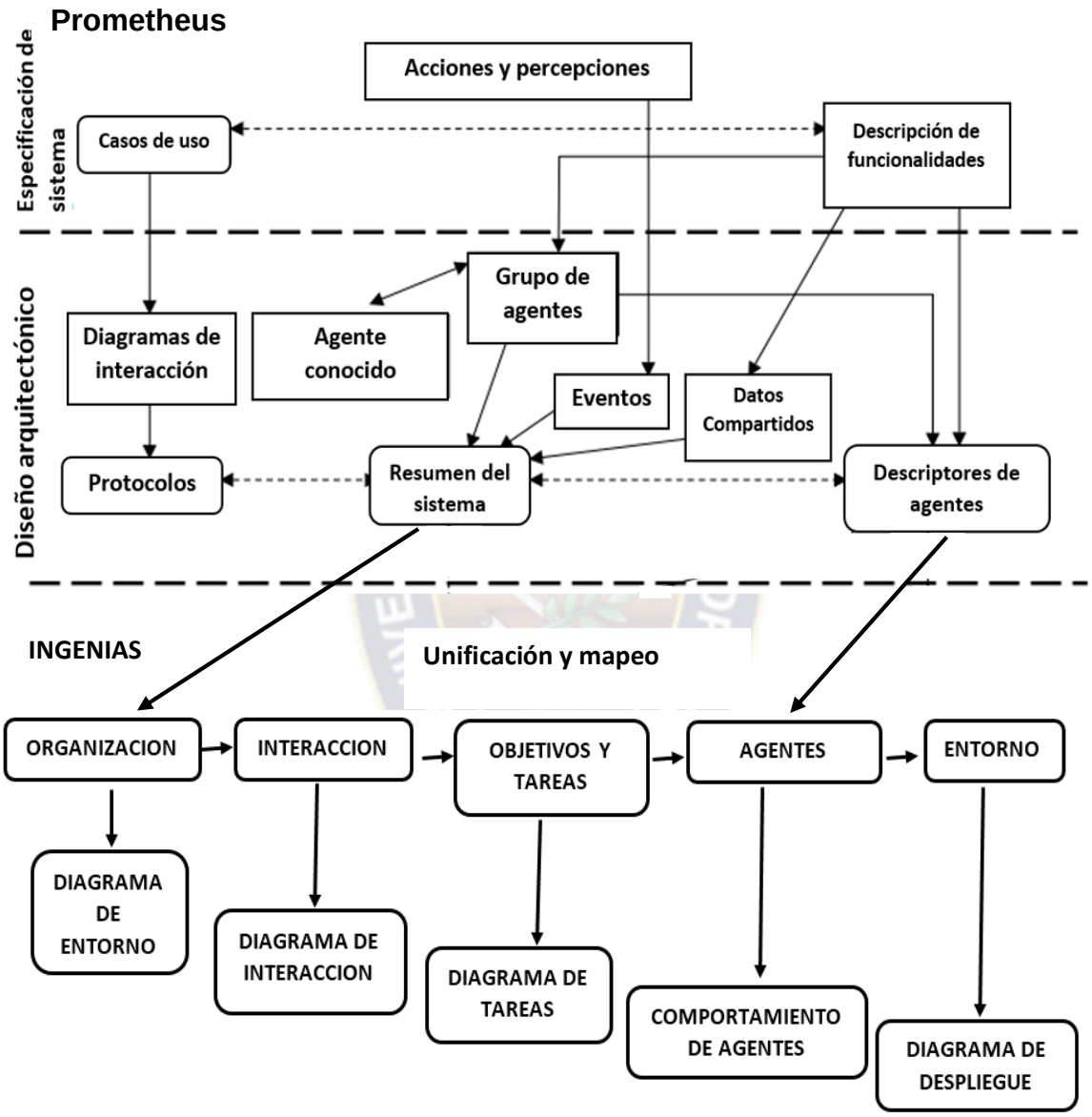


Figura 3.1. Unificación de las metodologías Prometheus - INGENIAS

En las tablas 3.1 y 3.2 en las cuales se muestra la comparación entre metodologías y herramientas respectivamente.

	PROMETHEUS	**INGENIAS**
Proceso de desarrollo adecuado	SI	NO: Basado en el USDP (análisis y fases de diseño). Un proceso ágil
Proceso general para generar código desde los modelos	NO: Sólo obtiene el código para el lenguaje JACK	SÍ: Basado en definiciones de plantilla
Proceso de desarrollo interactivo	si	si
Desarrollo orientado a modelos (MDD)	NO: Sólo se propone una correspondencia entre los modelos de diseño y JACK código	SI
Captura de requisitos	SI	SI
Meta-modelo	SI	SI
Mecanismos para descubrir Agentes e interacciones Entre los agentes	SÍ: Funcionalidades de grupos a través de Criterios de cohesión y acoplamiento	NO
Modelado de agente	Agentes similares a BDI	Agentes con estados mentales

Tabla 3.1. Comparando Prometheus e INGENIAS
Fuente:Fernández-Caballero, Gascueña, 2009.

En la tabla 3.1 se visualiza una comparación de ambas metodologías, en la tabla 3.2 se aprecia la comparación de albas metodologías apoyadas en sus propias herramientas y se puede apreciar que ambas tienen fortalezas distintas en su mayoría lo cual una vez combinadas ambas podrían enriquecer el análisis del sistema multi-agente.

	PDT	IDK
Metodología apoyada	**PROMETHEUS**	**INGENIAS**
La interfaz hace referencia al proceso de desarrollo	SÍ: Los diagramas se agrupan en tres niveles según los tres Fases de Prometheus	NO: Posibilidad de crear paquetes que corresponden a las diversas fases del proceso. modelos de cada fase se añaden al correspondiente paquete
Mecanismos para priorizar partes del proyecto	SÍ: Tres niveles de alcance (esencial, condicional y opcional)	NO
Generación de código	SÍ: JACK http: //www.agent - Software	SÍ: JADE http: //jade.tilab.com/
Generación de informes de la especificación MAS en HTML	SI	SI
Modelo de fragmentación en varias piezas	NO: Por ejemplo, sólo un diagrama puede ser creado para reunir todos los objetivos del sistema	SI
Guardar un diagrama como una imagen	SÍ: Formato .png. la resolución de la imagen puede ser configurado	SÍ: Formatos .jpg. .svg, .png, Wbmp, .bmp y .jpeg
Comunicación del agente	Definido en base de mensajes y protocolos de interacción. No se usa un lenguaje de comunicación específico. Para JACK, hay un módulo compatible Con FIPA	Definido de acuerdo con actos de comunicación del lenguaje de comunicación del agente (ACL) propuesta por el FIPA http://www.fipa.org/specs/fipa00061 /
Utilidad para simular MAS especificaciones antes generando el código final	NO	SÍ: Realizado en la plataforma JADE. Es posible gestionar la interacción tareas, y para inspeccionar y modificar la capacidad mental de los estados de agentes
Planos ejecutables como autónomo	Descripción textual SI	Descripción gráfica SÍ
Integración en Eclipse	SÍ: Véase [14]	SÍ: en versión IDK 2.7

Tabla 3.2. Comparación de PDT e IDK
Fuente: Fernández-Caballero, Gascueña, 2009.

El problema se acoto considerablemente puesto que se pretende realizar un estudio y proponer una solución mediante el modelo de un Sistema Multi-Agente que administre un test e identificar las acciones específicas que cada uno de sus agentes debe seguir para

cualificar conocimientos, para alcanzar este objetivo de manera exitosa basándonos en lo planteado con anterioridad

3.2. Especificación del sistema

A continuación, se muestra una gráfica la cual plasma lo que básicamente gestionaran los agentes de manera jerárquica con la base de preguntas el cual será nuestro objeto de estudio.

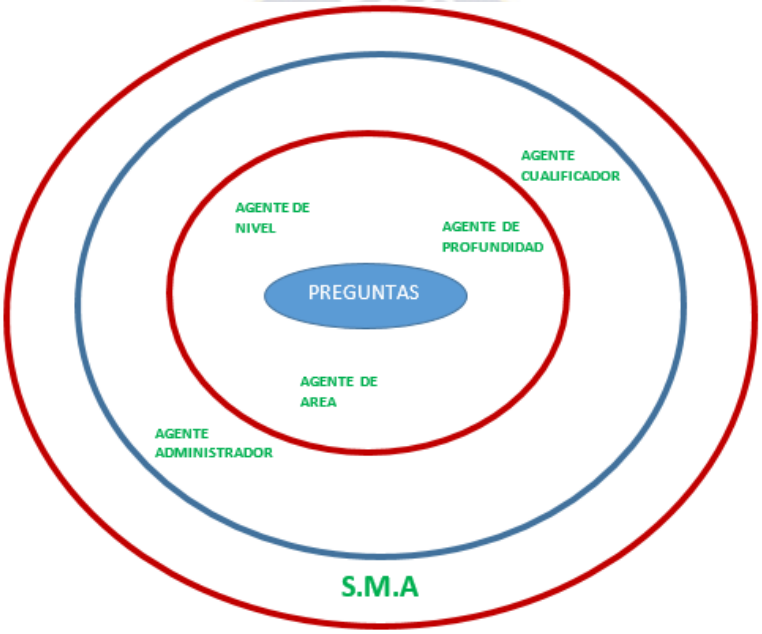


Figura 3.2. Sistema multi-agente en el test

En la figura 3.2 se muestra básicamente los niveles de jerarquía de los agentes dentro del modelo, en la figura 3.3. Se muestra como los agentes de área, nivel y profundidad, controlaran las diferentes partes del modelo de banco de preguntas propuesto, el estudio de preguntas que contendrá este banco no será contemplado puesto que se requiere un equipo de profesionales dedicados al área para poder establecer preguntas que en el mejor de los casos serian reactivos de respuesta al ítem creados de manera especial, pero se vio por

conveniente aferrarse al modelo de reactivos de opción múltiple expuesto en el anterior capitulo.

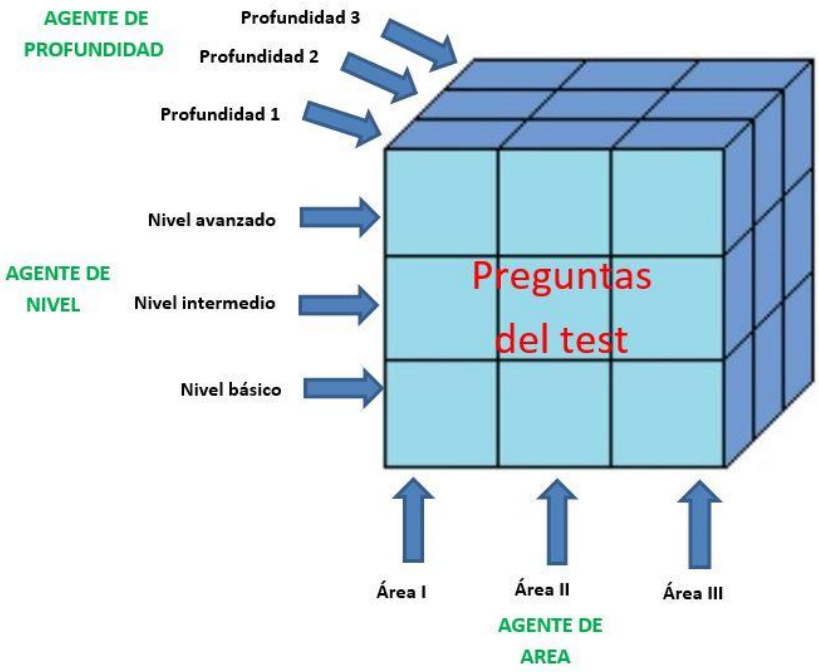


Figura 3.3. Modelo multidimensional de preguntas gestionado por los agentes inteligentes

Con este modelo se pretende mejorar la extracción de conocimientos del usuario que se someta al test, las preguntas se seleccionan por área, pero para el presente documento usaremos un área (redes) para el prototipo, pero se modelara la interacción de este agente y además de este hecho se siguen clasificando las preguntas por niveles y profundidad en cada nivel, los agentes se encargaran de gestionar el modelo propuesto de preguntas según lo descrito a continuación.

Inicialmente el estudiante en este caso llamado (usuario) ingresa al sistema con un *login* y *password* previamente registrados, el sistema inicia el test con preguntas básicas de cada una de las áreas.

El sistema analiza las respuestas del usuario después de cada pregunta realizada y asigna las siguientes preguntas en función de lo respondido por el usuario, subiendo así el nivel de las preguntas dependiendo de las respuestas ofrecidas a cada área, al mismo tiempo, el sistema profundiza las preguntas de cada nivel pertenecientes a las o el área que el usuario tenga mejor desempeño.

El sistema interactúa con el usuario en función de las respuestas subiendo o bajando el nivel en algunas áreas y lo mismo con la profundidad todo esto dependiendo de las respuestas efectuadas por el usuario, dándose el caso en el que no se responda asertivamente las respuestas el sistema realizará solo preguntas del nivel básico y en determinado momento dará por terminado el test.

Finalmente, el test termina cuando el sistema determina que se vieron agotadas todas las posibilidades de extracción mediante las respuestas al test por parte del usuario, El sistema ofrecerá un veredicto del test realizado, el cual especificará el grado de dominio del área y el nivel obtenido dentro del test, esta acción la realizará para todas las áreas contempladas en el test.

3.2.1. Identificación de actores

Para el desarrollo de la investigación y el desarrollo de agente inteligente es necesario la identificación de los posibles usuarios o actores que interactúan en el SMA y la especificación de los mismos estos actores deben ser posibles de identificar en cualquier sistema que quiere utilizar al agente. Al agente inteligente que se desarrollará se lo dividió en las dos funciones principales que se realizará, la primera será la interacción con los agentes y la identificación de los actores la cual será descrita a continuación.

ACTOR	CARACTERISTICAS
Actor Usuario (acUsuario)	• No se comunica directamente con el agente, solo será comunicación directa con el sistema. • Este actor será el que alimente al sistema mediante las respuestas que este brinde a las respuestas.
Actor ControlNivel (acControlNivel)	• Se encarga de analizar y asignar qué nivel de preguntas corresponde al usuario según el avance del mismo
Actor ControlProfundidad (acControlProfundidad)	• Controlará la profundidad de nivel de las preguntas del test • Analizara el salto de preguntas en profundidad correspondiente al nivel.
Actor ControlÁrea (acControlÁrea)	• Se encargará de gestionar las áreas a profundizar en el test. • Se encarga de que los órdenes de las preguntas no se repitan respetando los niveles de estas.
Actor Cualificador (acCualificador)	• Se encargará de analizar los resultados de los agentes (nivel, profundidad, área). • Diferenciara el área en las cuales se desenvuelve mejor el usuario con las preguntas. • Brindará un veredicto ante lo analizado en el test.
Actor Administrador (acAdministrador)	• Se encargará de gestionar la continuidad y la finalización del test. • Responsable de la entrega de la matriz de resultados para la cualificación.
Actor ControlPregunta (acControlPregunta)	• Controlará el número de respuestas fallidas y acertadas del usuario • Controlará la posición de las preguntas • Se encargara de guardar las respuestas en la matiz de respuestas
Actor AsignaPregunta (acAsignaPregunta)	• Se encargará de extraer las preguntas del test. • Se responsabilizara de realizar un incremento o decremento en los niveles y profundidades de las preguntas

Tabla 3.3: de actores y sus respectivas características

En este caso de estudio se utilizará una nomenclatura para nombrar cada tipo de entidad usada en el sistema, en la tabla 3.4 se observa el nombre de cada tipo de entidad tiene como prefijo unas iniciales que identifican cual es el tipo al que pertenecen.

Entidad	Identificador
Actor	acNombreDelActor
Agente	AgNombreDelAgente
Percepción	PNombreDelaPercepción
Acción	ANombreDeLaAcción

Tabla 3.4. Descripción de entidades

En el diagrama (figura 3.4.) se muestra un esbozo de las interacciones que se producen entre el sistema y su entorno para este punto de identificaron los actores ya mencionados en la tabla 3.3.

El actor ***acUsuario*** representa al sujeto en nuestro caso el estudiante que se somete a la prueba, este actor se encargará de responder a las preguntas del test, esta acción ***pRespuesta*** será la percepción a la que responderán los demás actores.

El *actor* ***acAdministrador*** representa el agente que inicia el sistema y se encarga de gestionar el normal avance del test o la finalización del mismo, además se encarga también de la entrega de la matriz de respuestas para la respectiva una vez haya finalizado el test cualificación estos procesos se describen en los escenarios ***ControlPregunta Scenario***

El actor ***acControlPregunta*** representa el agente que controla las preguntas del test recibe la percepción ***pRespuesta*** que son las respuestas, este proceso se describe en los escenarios *validación* ***Scenario*** *y* ***ControlPregunta Scenario***, ***ControlArea Scenario***, ***ControlNivel Scenario*** **y** ***ControlProofundidad Scenario***.

El actor ***acAsignaPregunta*** representa el actor que se encargará de asignar las preguntas del test, este actor recibirá la posición de la pregunta anterior y la respuesta validada para asignar la siguiente pregunta en el test.

El actor ***acControlArea*** representa al agente que controla las áreas de las preguntas en el test, este agente recibirá la respuesta validada del actor ***acUsuario***, cada área está dividida en niveles, básico intermedio, avanzado.

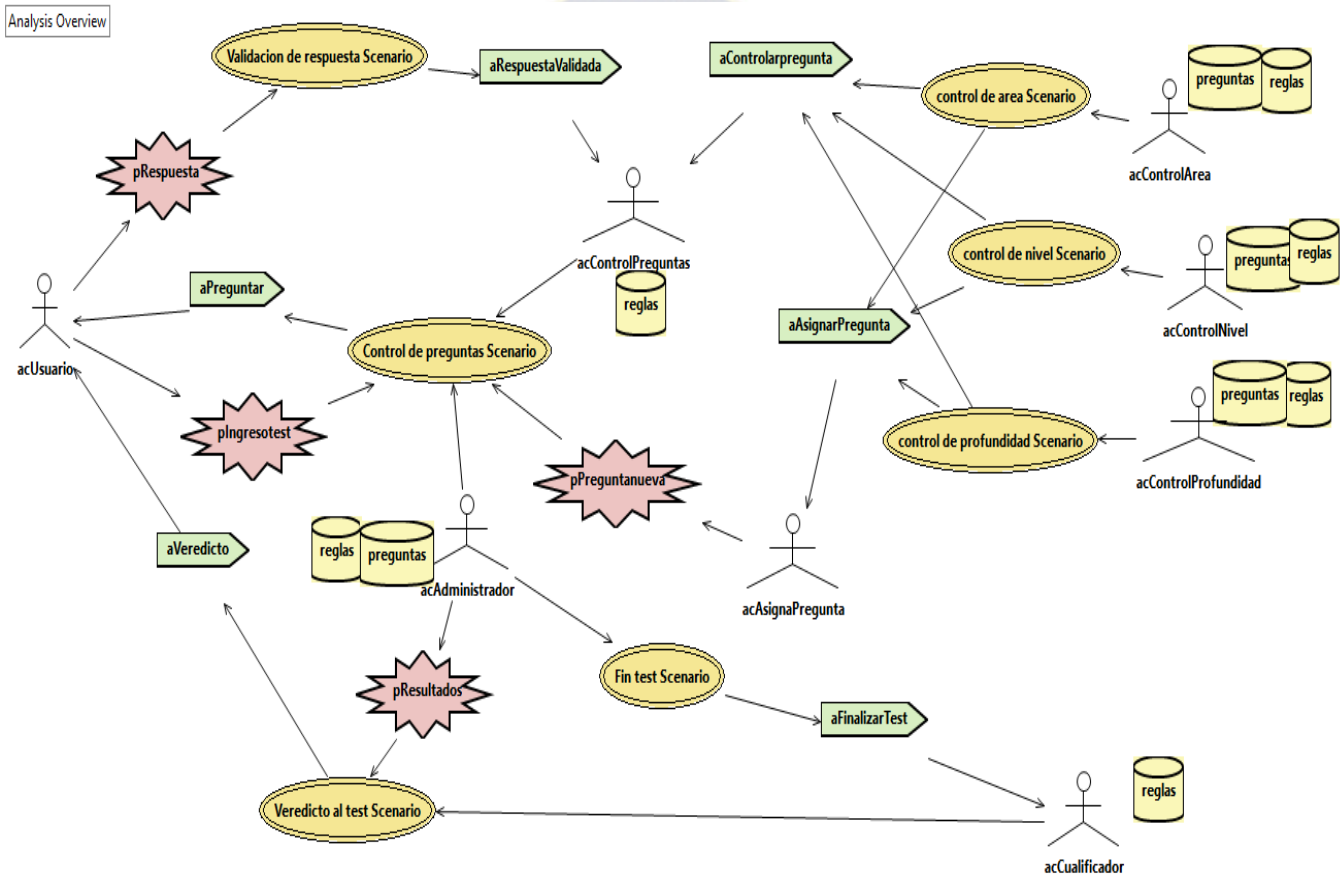


Figura 3.4. Diagrama de visión general del análisis

El actor ***acControlNivel*** controlara los niveles de las preguntas en el test, así mismo cada nivel estará dividido por subniveles a los cuales se llamará profundidad, es decir que cada nivel tendrá una profundidad de nivel la cual ira incrementando la dificultad de las

preguntas a medida que más respuestas positivas se reciban. El **actor** ***acControlProfundidad*** será el encargado de controlar la profundidad de preguntas en cada nivel, estos procesos se describen en los escenarios ***ControlArea Scenario***, ***ControlNivel Scenario*** y ***ControlProofundidad Scenario***.

El ***acCualificador*** será el encargado de brindar un veredicto con respecto a las respuestas acertadas y a los niveles y profundidad de nivel obtenido en cada área, estos procesos se describen en los escenarios ***fin test scenario*** y ***veredicto al test scenario***.

3.2.2. Desarrollo de escenarios

En este proceso de la metodología de desarrollo para agentes inteligentes se detallarán los escenarios que participarán en el sistema por lo cual se mostrara cada escenario del sistema, en la figura 3.5 se muestran los escenarios encontrados en el sistema con una breve explicación de los mismos en el rectángulo superior.

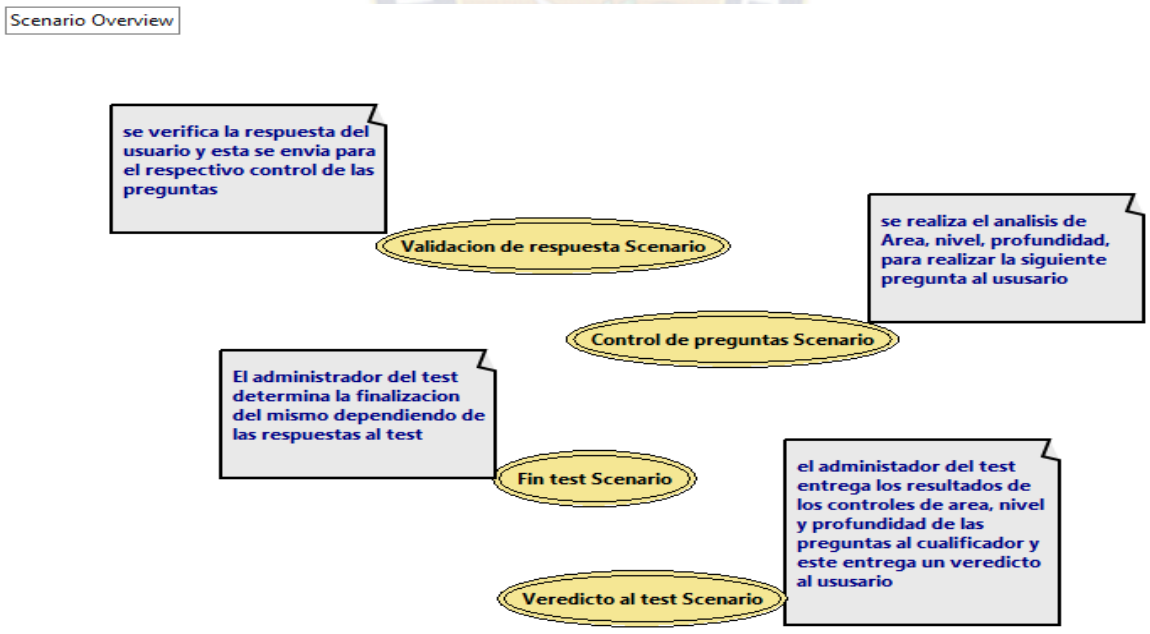


Figura 3.5. Diagrama de descripción general de escenarios

Un escenario es una secuencia de pasos estos forman un guion que describen un posible comportamiento del sistema frente a esa situación, los escenarios principales como ser ***Validacion de respuesta Scenario*** y ***Control de preguntas Sccenario*** describen el flujo de la realización de las preguntas y la resolución a las respuestas junto con el control de cada una de estas en la base de preguntas respectivamente. El primeo describe la situación en la que **el *acUsuario*** responde a la pregunta que le es realizada y esta respuesta es validada para ser trasladada a ***Control de preguntas Sccenario*** en esta fase se controlan las preguntas que se realizaran estas preguntas se analizan con más detalle empezando con el área de la cual saldrá las preguntas, en ***Control Nivel Scenario*** se verifica en que área se encuentra la pregunta lanzada con anterioridad y el flujo de áreas las cuales se van incursionando a medida que avanza el test estas áreas también están clasificadas por nivel de conocimiento entonces no se harán preguntas a ciertas áreas sin que se tenga el nivel requerido en las áreas de niveles primarios, estos flujos nos trasladan al siguiente escenario el cual es ***ControlNivel Scenario*** en el cual se controla el nivel actual del usuario en base a las preguntas respondidas al test, este escenario interactuar con los escenarios ***ControlNivel Scenario*** *y* ***ControlProfundidad Scenario***, puesto que dependiendo de la profundidad que se encuentre en el nivel se puede dar un incremento o decremento el mismo, en el ***ControlProfundidad Scenario*** se determina la profundidad dentro de un nivel determinado dependiendo a la respuestas la profundidad en las preguntas puede avanzar o cambiar de nivel o área.

Finalmente, ***FinTest Scenario*** el sistema determina en función al avance y evolución del test si este termina o prosigue en el caso de que se determine que el test concluyo se manda los resultados del test para la cualificación, en el ***Veredicto al Test Scenario*** se realiza el análisis de los resultados obtenidos en cada área evaluada y se entrega una sentencia al usuario con respecto a los conocimientos demostrados.

3.2.3. Definición de metas

Los escenarios descritos con anterioridad están directamente relacionados con los objetivos del sistema, es decir que por cada escenario existe un objetivo relacionado, entonces la consecución de dicho objetivo se cumple cuando se han cumplido todos los pasos que componen la asociación del escenario. En la tabla 3.5 se describen los objetivos del sistema

Tipos de Meta	Descripción
Meta principal	Gestionar las preguntas del test de manera que exista el mayor porcentaje de exactitud en la cualificación de conocimientos. (veredicto al test)
Sub-metas	• Validación de respuestas. • Control de respuestas • Almacenamiento del resultado de las respuestas y marcado de las misma en la base de preguntas. • Control de preguntas a ser efectuadas al usuario. • Control de áreas preguntadas y o a ser realizadas al usuario. • Control de nivel de preguntas realizadas o a ser realizadas al usuario. • Control de profundidad de preguntas correspondientes al nivel en el que el usuario se encuentre. • Gestión y selección de preguntas dadas al usuario. • Actualización de estados en la base de preguntas. • Gestión de sectores en la base de preguntas. • Gestión en el proceso de conclusión del test • Finalización del test • Continuidad del test • Termino del test

Tabla 3.5 descripción general de objetivos del sistema

En la figura 3.6 se muestra la descripción general de los objetivos o metas del sistema, descritos en la tabla 3.5

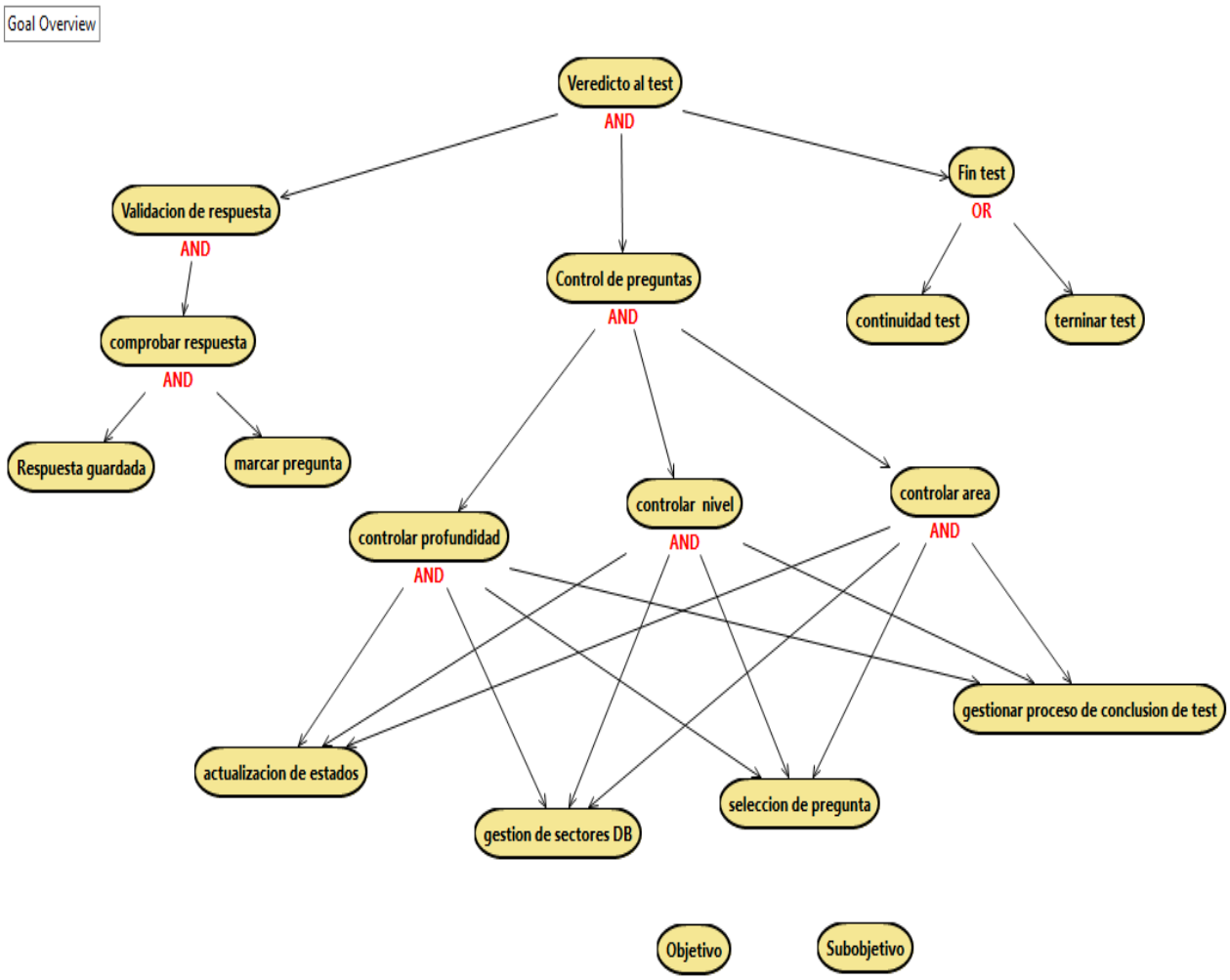


Figura 3.6. Diagrama de descripción general de objetivos del sistema

3.2.4. Descripción de roles

Los roles del sistema se obtienen de la agrupación de objetivos, la relación de acciones y percepciones, como resultado de estas agrupaciones se identifican roles para cada una de estas funcionalidades que el sistema debe soportar (figura 3.7).

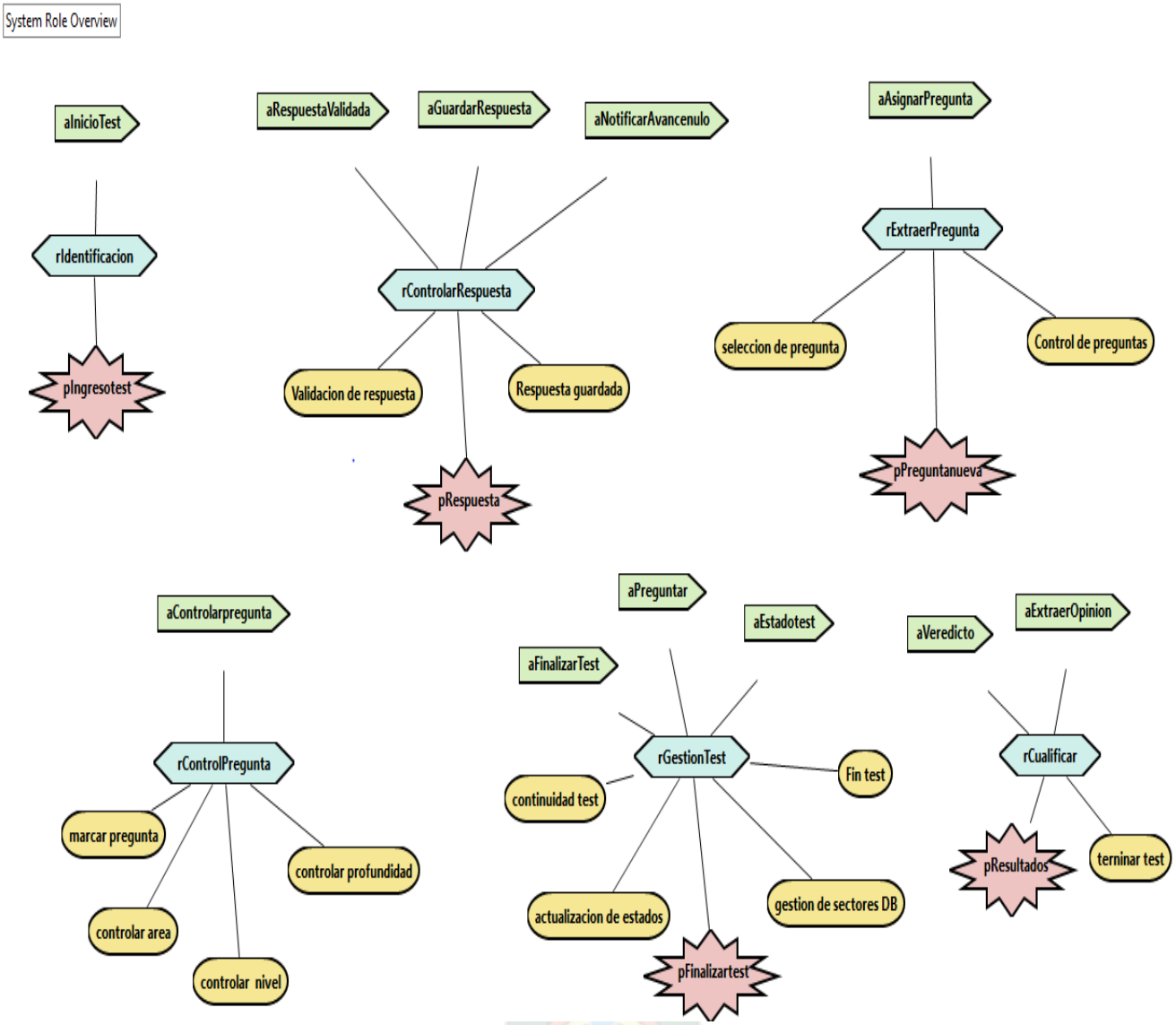


Figura 3.7. Diagrama de roles del sistema

En la tabla 3.6. Se describen brevemente los roles identificados.

Rol	**Descripción**
rIdentificacion	Gestionará el acceso del usuario al sistema
rControlarespuesta	Responsable de la validación de las respuestas, su objetivo es corroborar las respuestas y guardarlas

rExtraerpregunta	Su objetivo es extraer las preguntas del banco de preguntas
rControlpregunta	Responsable del control del estado y nivel de las preguntas en la base que son entregadas al usuario
rGestionTest	Gestiona la continuidad del test o de la finalización del mismo
rCualificar	Responsable de analizar las respuestas y entregar el veredicto al usuario

Tabla 3.6. Descripción de los roles

3.3. Arquitectura del sistema

3.3.1. Determinación de tipos de agentes

Para la determinación de los tipos de agentes tendrán que ser verificados los roles principales, una vez identificados los roles en la fase de identificación del sistema, la primera cuestión que hay que resolver en la fase de diseño arquitectónicos identificar los tipos de agentes que van a formar parte del sistema. Para este propósito, se toma como base los roles del sistema y se agrupa de tal modo que cada agente introducido sea responsable de administrar o llevar a cabo uno o varios roles.

En pocas palabras Cada rol se especificó según el tipo de acción que realizara, esto es necesario pues se identificará cuantos tipos de agentes son necesarios en el sistema a desarrollar

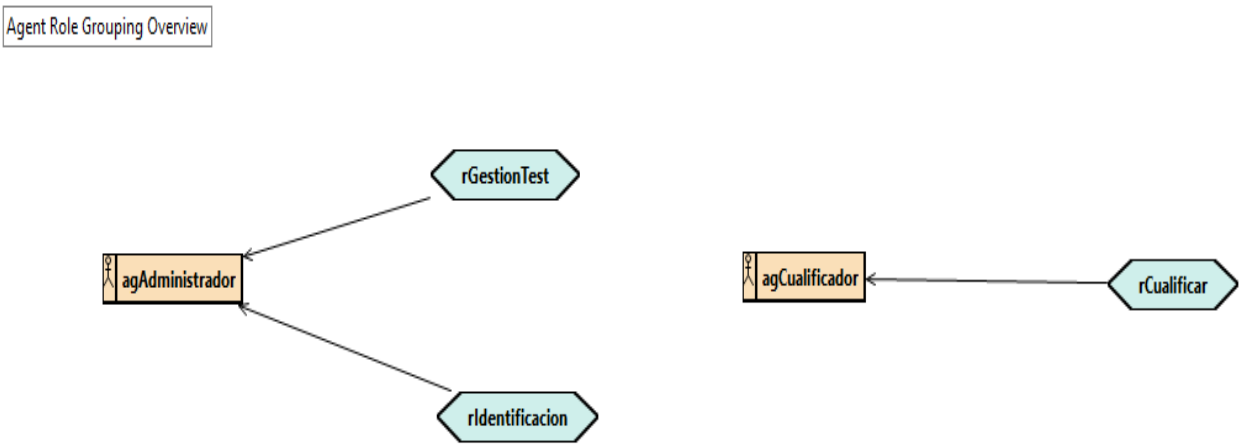


Figura 3.8 A. Diagrama general de agrupación por rol

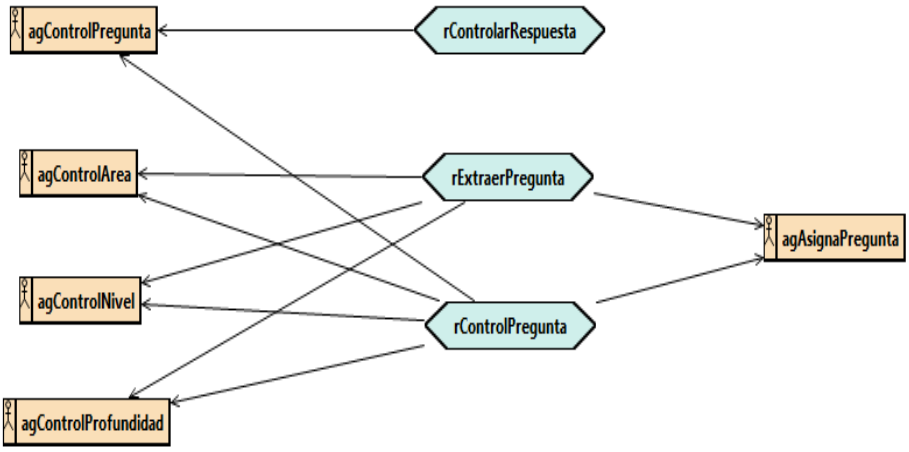


Figura 3.8 B. Diagrama general de agrupación por rol

Luego de realizada esta agrupación se requiere realizar otro tipo de agrupación por los datos en nuestro caso las preguntas del test que hacen referencia a la base de datos, pues este se involucra con el sistema y los agentes.

Una vez analizadas preguntas en la base de datos se podrán agrupar estas en los roles a los cuales los agentes pertenecen, al mismo tiempo en esta etapa del proceso de la metodología de desarrollo se definirá al rol según el tipo de pregunta en la base de datos previa validación.

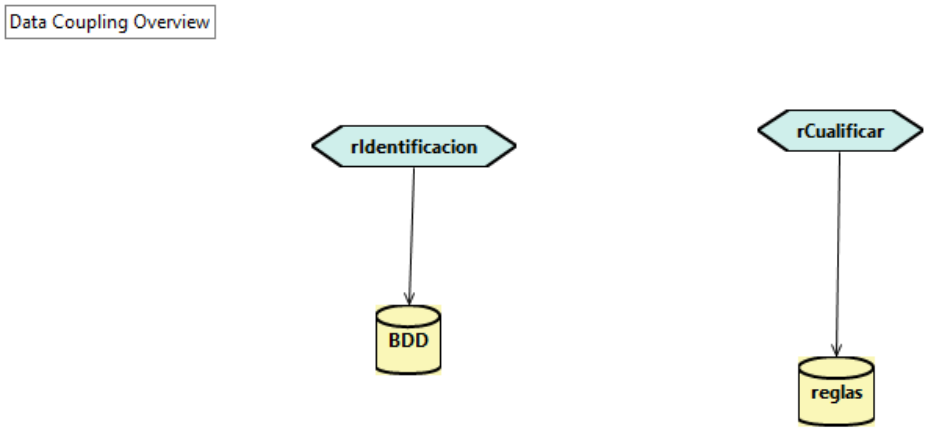


Figura 3.9 A. Diagrama general de agrupación por datos

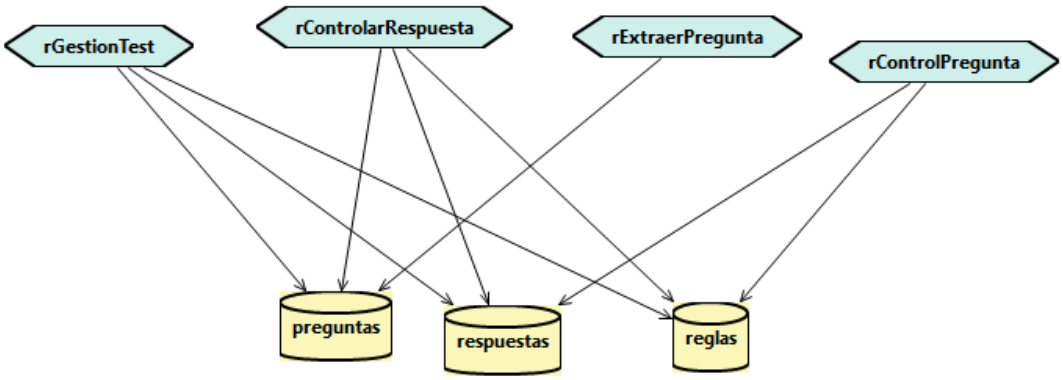


Figura 3.9 B. Diagrama general de agrupación por datos

3.3.2. Análisis descriptivo general

En esta parte o etapa de la metodología empleada se realizará un análisis general de cómo será el sistema y la interacción del mismo con los agentes necesarios para una mejora en el proceso de evaluación haciendo uso de la administración del test creado. Para la realización de este análisis se hará una descripción minuciosa de las iteraciones y los componentes del sistema.

Para las relaciones existentes en el diagrama generado en este punto o paso de la metodología Prometheus, se tiene que describir cada elemento y las relaciones entre los mismos, la relación entre los usuarios y los escenarios del sistema por los cuales se tendrán acceso a las preguntas del test las cuales serán proporcionadas por el agente según las respuestas que se tenga del usuario, estas respuestas serán los estímulos que el agente recibirá para interactuar con el sistema y la base de preguntas para tomar las decisiones designadas.

Una vez procesados estos estímulos brindados por el usuario y tomando las acciones correspondientes a cada uno de estos, el agente se comunicará con la base de preguntas del test mediante los protocolos establecidos para realizar la acción correspondiente a los estímulos que se presenten en las respuestas mediante el test.

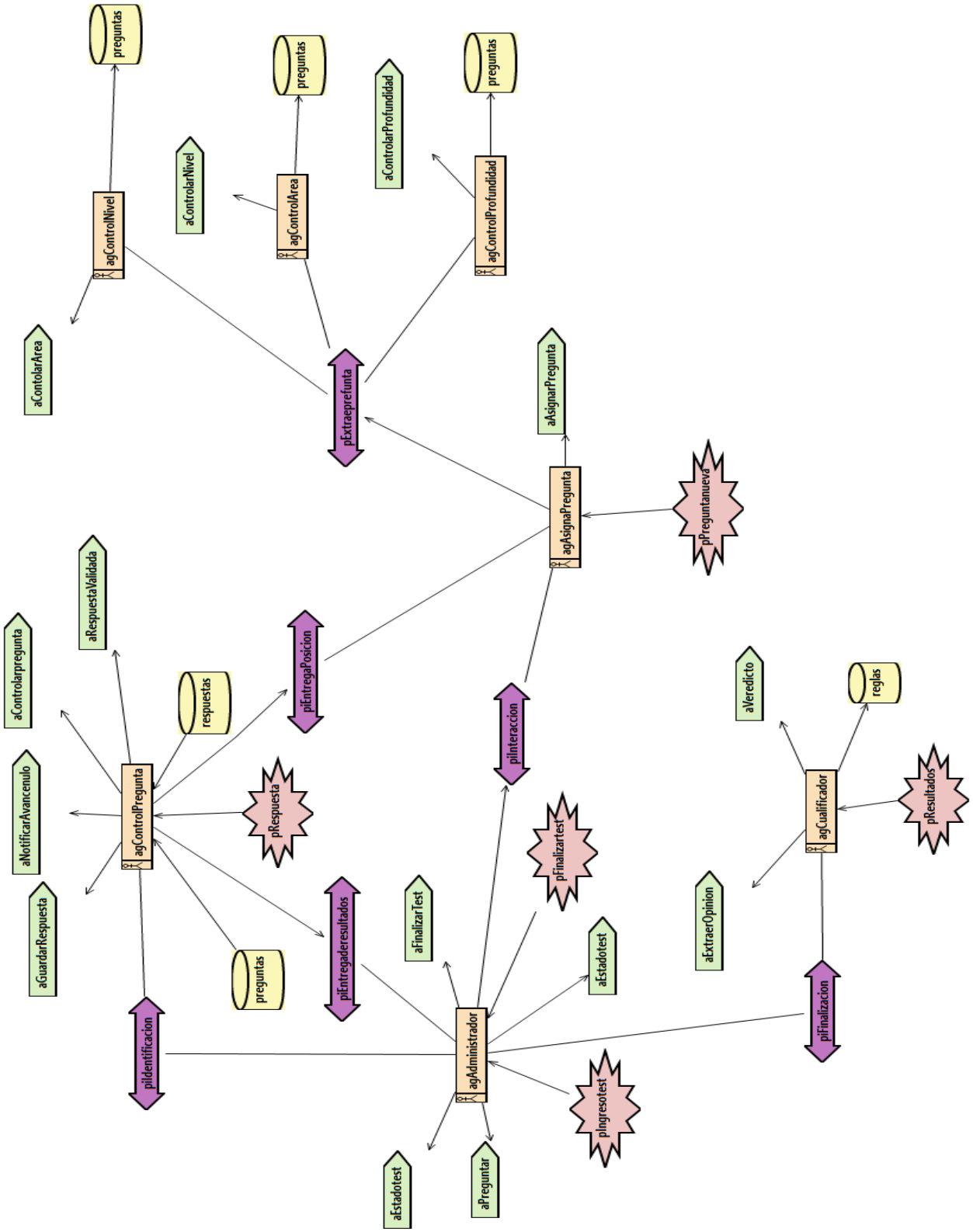


Figura 3.10. Diagrama de visión general del sistema

En la figura 3.10 se establece la representación de las relaciones existentes en el sistema.

3.4. Diseño detallado del sistema

3.4.1. Transformación de estructuras Prometheus a Ingenias

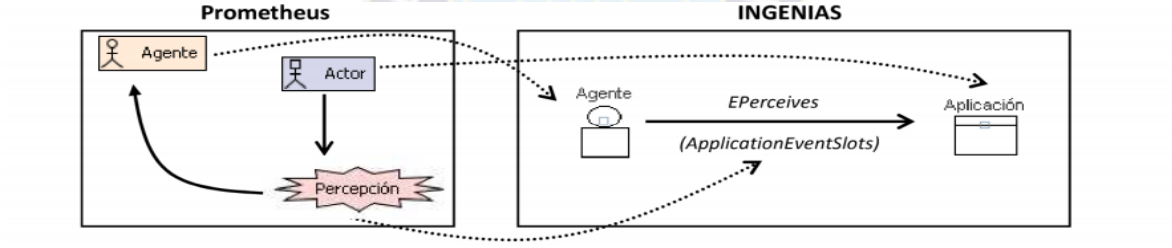
La fase de diseño detallado requiere, primeramente, la transformación de los conceptos de Prometheus incluidos en el diagrama de visión general del sistema en conceptos equivalentes en la metodología INGENIAS. Puesto que ambas metodologías no utilizan la misma terminología para describir los elementos que conforman el sistema. Las equivalencias entre las entidades del lenguaje de modelado de Prometheus y el lenguaje de modelado de INGENIAS se pueden resumir a continuación:

- La entidad agente está presente en Prometheus e INGENIAS y por lo tanto su traducción es directa.
- Un actor en Prometheus queda descrito en INGENIAS mediante una aplicación con la que interacciona un agente por medio de métodos y eventos.
- Una entidad aplicación, presente en INGENIAS, encapsula una serie de métodos y eventos y se relaciona con uno o varios agentes. Un método es el procedimiento que describe una acción, que puede ser invocado junto con una serie de argumentos y devolver un resultado. El método puede enviar un evento de aplicación, el cual puede contener información (figuras 3.11 y 3.13), a un agente.
- La interacción entre agentes y aplicaciones se realizan a través de métodos. De esta manera, cada actor de Prometheus se ha llevado a INGENIAS transformado en una aplicación, dentro de la cual se creará un método por cada acción que un agente puede ejecutar sobre este actor y, seguidamente, se ha creado un evento de aplicación por cada percepción que puede recibir un agente de este actor (figurass 3.11 y 3.12).
- Respecto a las interacciones entre agentes se resalta que los mensajes utilizados en Prometheus se reflejan en INGENIAS mediante las denominadas Unidades de

Interacción, equivalentes a los mensajes, las cuales pueden contener información adicional, al igual que sucede en los eventos de aplicación (figura 3.13).

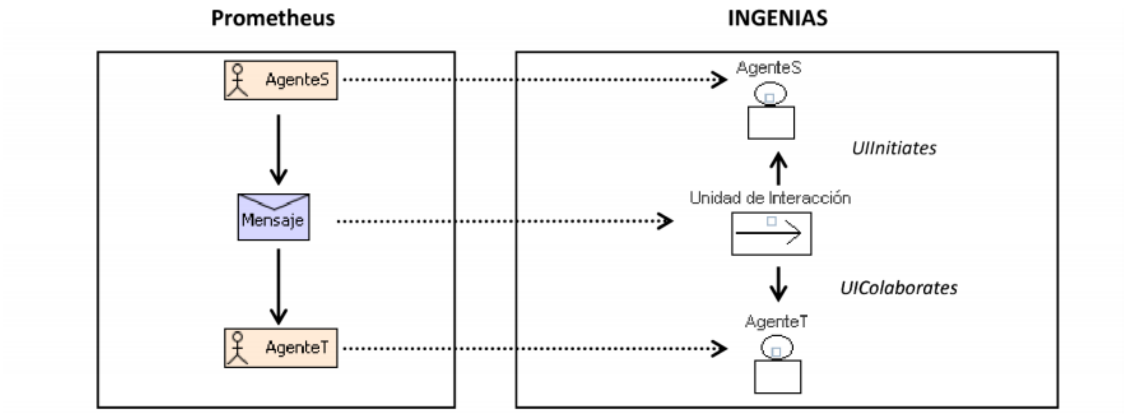
- Por último, los datos en Prometheus no tienen una equivalencia directa en INGENIAS. Por lo tanto, la manera de reflejar un dato se hace mediante una aplicación con la que pueden interaccionar los agentes (figura 3.13).

Las figuras 3.11, 3.12 y 3.13 muestran gráficamente la transformación de conceptos que debe realizarse al traducir un modelo de Prometheus a INGENIAS. Cada figura muestra en su parte izquierda el modelo original en Prometheus y en su parte derecha el modelo equivalente en INGENIAS. En la parte inferior de cada figura aparece una tabla que indica los diagramas que se ven afectados en la transformación. En la columna de la izquierda aparece lo que afecta en Prometheus y en la de la derecha lo que afecta a INGENIAS.



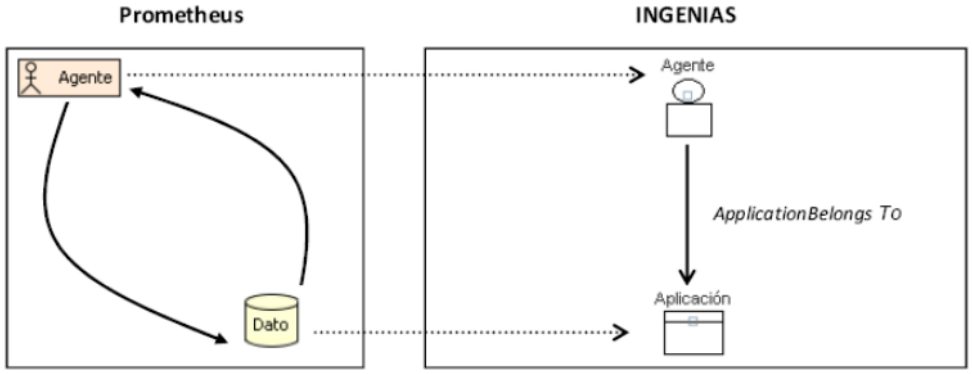
Prometheus	INGENIAS
Agente ← Percepción ← Actor	Modelo del entorno Agente – *EPerceives* → Aplicación
- Descriptor Percepción: campo *Information carried* - Diagrama de visión general del sistema: Agente ← Percepción - Protocolo de interacción: Percepción ← Actor	- Tipo de evento *ApplicationEventSlots* asociado a la relación EPerceives si Percept contiene información. - Tipo de evento *ApplicationEvent* asociado a la relación EPerceives si Percept no contiene información

Figura 3.11. Transformación de una percepción
Fuente: Basado en Prometheus and INGENIAS Agent Methodologies: A Complementary Approach



Prometheus	INGENIAS
AgenteS → Mensaje Mensaje ← → AgenteT Diagrama de visión general del sistema Protocolo de interacción - AgentS es el agente que envía el mensaje - AgentT es el agente que recibe el mensaje	Modelo de interacción: AgenteS ← *UlInitiates* – Unidad de Interacción Unidad de Interacción ← *UIColaborates* – AgenteT - La Unidad de Interacción incluye un FrameFact si el mensaje contiene información

Figura 3.12. Transformación de un mensaje
Fuente: Basado en Prometheus and INGENIAS Agent Methodologies: A Complementary Approach



Prometheus	INGENIAS
Agente ← → Dato - Diagrama de visión general del sistema: Agent ← → Data El dato es de granularidad gruesa	Modelo del entorno: Agente – *ApplicationBelongs To* → *Aplicación*

Figura 3.13. Transformación de un dato
Fuente: Basado en Prometheus and INGENIAS Agent Methodologies: A Complementary Approach

En esta etapa de la metodología de desarrollo de agentes inteligentes se realizará una descripción detallada de las percepciones y acciones vinculadas a cada agente inteligente para poder cumplir con los objetivos y metas para los que fue diseñado. Por lo cual se asocia a cada agente con las percepciones que este recibirá y también con las acciones que este agente ejecutará para especificar que percepción le corresponde a cada una de las acciones.

El resultado de estas transformaciones se resume en tabla 3.7 la cual muestra en su lado izquierdo las entidades procedentes de Prometheus junto a sus figuras asociadas, y en el lado derecho, las entidades equivalentes en INGENIAS y sus respectivas figuras.

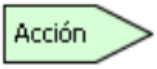
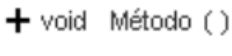
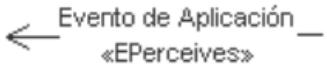
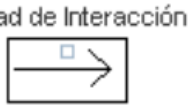
Prometheus		**INGENIAS**	
Entidad	**Figura**	**Entidad**	**Figura**
Agente		Agente	
Actor		Aplicación	
Acción		Método	
Percepción		Evento de Aplicación	
Mensaje		Unidad de Interacción	
Dato		Aplicación	

Tabla 3.7. Equivalencias de conceptos entre Prometheus e INGENIAS
Fuente: Basado en Prometheus and INGENIAS Agent Methodologies: A Complementary Approach

A continuación, se visualizarán una serie de tablas para efectivamente mostrar la transformación de conceptos entre las dos metodologías, en la parte izquierda se muestra la entidad Prometheus y su tipo (origen) y en la derecha la entidad Ingenias y su tipo respectivamente.

Prometheus		**INGENIAS**	
Entidad	**Tipo**	**Entidad**	**Tipo**
agAdministrador	Agente	*agAdministrador*	Agente
agControlPreguntas	Agente	*agControlPreguntas*	Agente
agControlNivel	Agente	*agControlNivel*	Agente
agControlArea	Agente	*agControlArea*	Agente
agControlProfundidad	Agente	*agControlProfundidad*	Agente
agCualificador	Agente	*agCualificador*	Agente
agAsignaPreguntas	Agente	*agAsignaPreguntas*	Agente
agVerificaRespuesta	Agente	*agVerificaRespuesta*	Agente

Tabla 3.8. Agentes obtenidos desde la metodología Prometheus

Se puede apreciar que los agentes que se usaran en la metodología INGENIAS, son los obtenidos en la metodología Prometheus

Prometheus		**INGENIAS**	
Entidad	**Tipo**	**Entidad**	**Tipo**
acControlPreguntas	Actor	acControlPreguntas	Aplicación
acAsignacionPregunta	Actor	acAsignacionPregunta	Aplicación
acAdministracion	Actor	acAdministracion	Aplicación
acCualificador	Actor	acCualificador	Aplicacion

Tabla 3.9. Aplicaciones obtenidas a partir de los Actores en Prometheus

Las tablas 3.11 a la tabla 3.14 describen las aplicaciones obtenidas en la trasformación de la metodología Prometheus obtenidos previamente. En la columna tipo se ve la transformación de la entidad de origen y la transformación y una descripción breve del contenido de cada entidad, se debe aclarar que los nombres obtenidos en la metodología Ingenias llega a ser la misma que se obtuvo en Prometheus, a continuación, se muestra un resumen del significado de cada uno de los tipos empleados (tabla 3.10).

Tipo	Descripción
P	Percepción
ME	Método
M	Mensaje
IU	Unidad de interacción sin información
IUFF	Unidad de interacción con información
AE	Evento de aplicación sin información
AES	Evento de aplicación con información
AC	Acción

Tabla 3.10. Leyenda
Fuente: Pavón, J., Gómez-Sanz, J. 2006.

Prometheus		**INGENIAS**	
Entidad	**Tipo**	**Tipo**	**Descripción**
pRespuesta	P	AES	Evento que contiene respuesta del usuario
aRespuestaValidada	AC	ME	Verifica la respuesta
aControlaPregunta	AC	ME	Controla la posición de la pregunta
aNotificarAvancenulo	AC	ME	Notifica sobre avance nulo o escaso en el test

Tabla 3.11. Aplicación ***acControlPregunta***

En la tabla 3.11 se muestra las funciones de la aplicación ***acControlPregunta*** la cual cuenta con un evento y tres métodos que a su vez son acciones que se toman dentro de la aplicación.

Prometheus		**INGENIAS**	
Entidad	**Tipo**	**Tipo**	**Descripción**
aAsignarPregunta	AC	ME	Revisa las posiciones para asignar una nueva pregunta
mPosicion	M		Mensaje que recibe la posición de la pregunta actual y la validación

Tabla 3.12. Aplicación ***acAsignaPregunta***

En la tabla 3.12 se aprecia que la aplicación ***acAsignaPregunta*** posee un método de acción y este se comunica a través de un mensaje recibido.

Prometheus		**INGENIAS**	
Entidad	**Tipo**	**Tipo**	**Descripción**
pResultado	P	AES	Recibe la base de resultados
aVeredicto	AC	ME	Proporciona al usuario una opinión de los conocimientos logrados en las diferentes áreas

Tabla 3.13. Aplicación ***acCualificador***

En la tabla 3.13 se aprecia que la aplicación ***acCualificador*** posee una percepción y un método de acción directa con el usuario.

Prometheus		**INGENIAS**	
Entidad	**Tipo**	**Tipo**	**Descripción**
pInicioTest	P	AE	Evento que indica que el usuario inicio una sesión en el sistema
pFinTest	P	AE	Evento que indica que el usuario finalizo el test
aEstadoTest	AC	ME	Notifica sobre avance del test
aPreguntar	AC	ME	Realiza la pregunta designada al usuario

Tabla 3.14. Aplicación ***acAdministrador***

En la tabla 3.14 se puede apreciar que la aplicación ***acAdministrador*** tiene dos percepciones y dos métodos de acción, cabe recalcar que a medida que se desarrolle el avance en esta parte de la metodología pueden aparecer nuevos métodos, mensajes, y percepciones pues poco a poco se hará más específico el modelado.

A continuación, se describe la aplicación obtenida en Ingenias al trasformar un dato de granularidad gruesa obtenido en el método Prometheus. Los datos BDD, Preguntas y Reglas, se transformaron en INGENIAS en una aplicación que permite gestionar el acceso a cada una de estas bases de datos que contiene información de los usuarios y preguntas del test.

Prometheus		**INGENIAS**		
Entidad	**Tipo**	**Entidad**	**Tipo**	**Descripción**
-	-	Comprobarusuario	ME	Comprueba en la BDD la validez de los datos de usuario introducido
-	-	Usuario correcto	AE	El usuario ingresa al sistema
.	.	Usuario incorrecto	AE	El usuario no ingresa al sistema

Tabla 3.15. Aplicación DBB

Las aplicaciones obtenidas darán como resultado el diagrama de entorno (figura 3.14) El diagrama incluye el conjunto de agentes y aplicaciones que existen en el sistema. Estas entidades y su relación están establecidas para reflejar la utilización de aplicaciones por parte de los agentes.

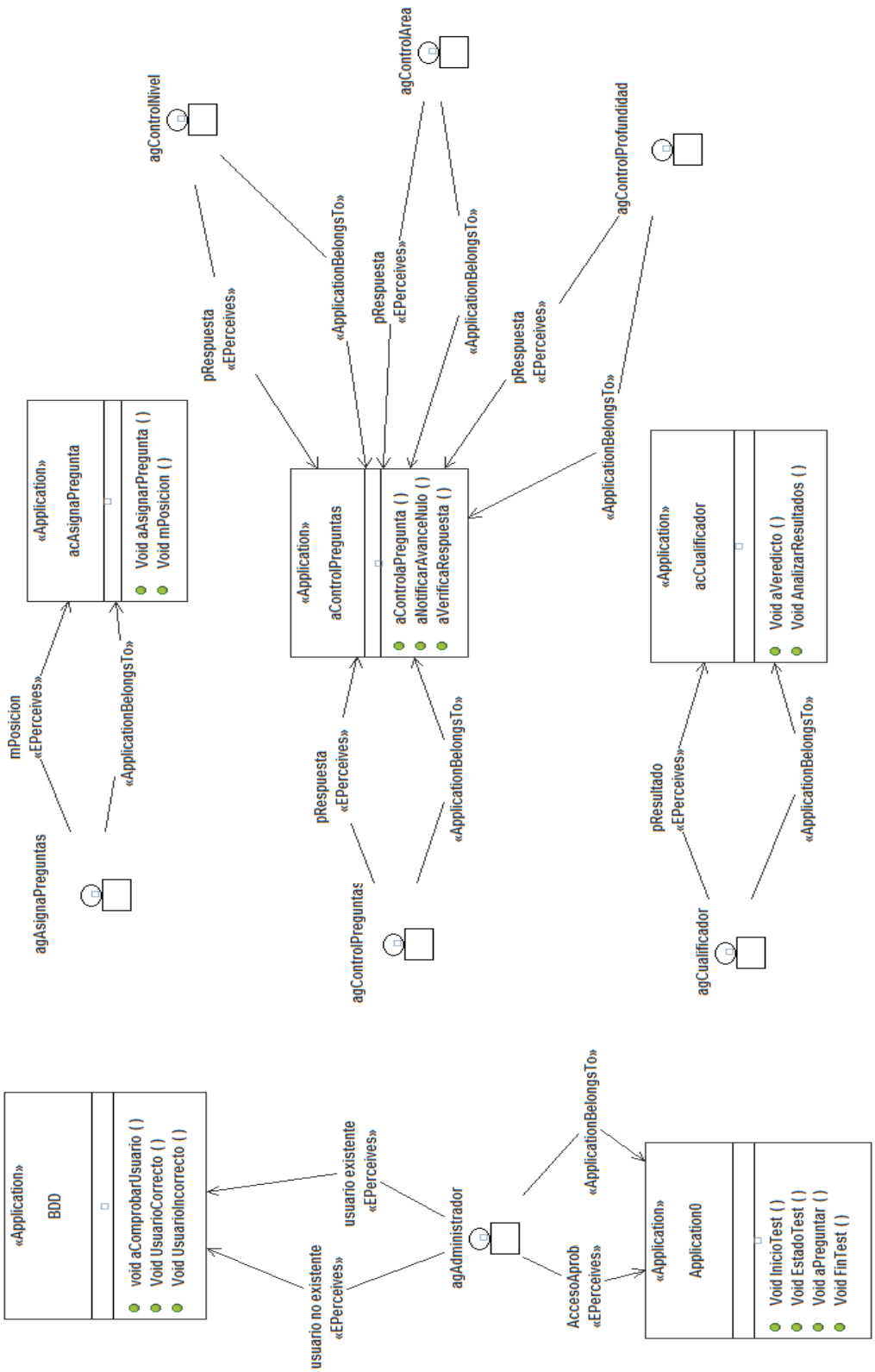


Figura 3.14A Diagrama de Entorno

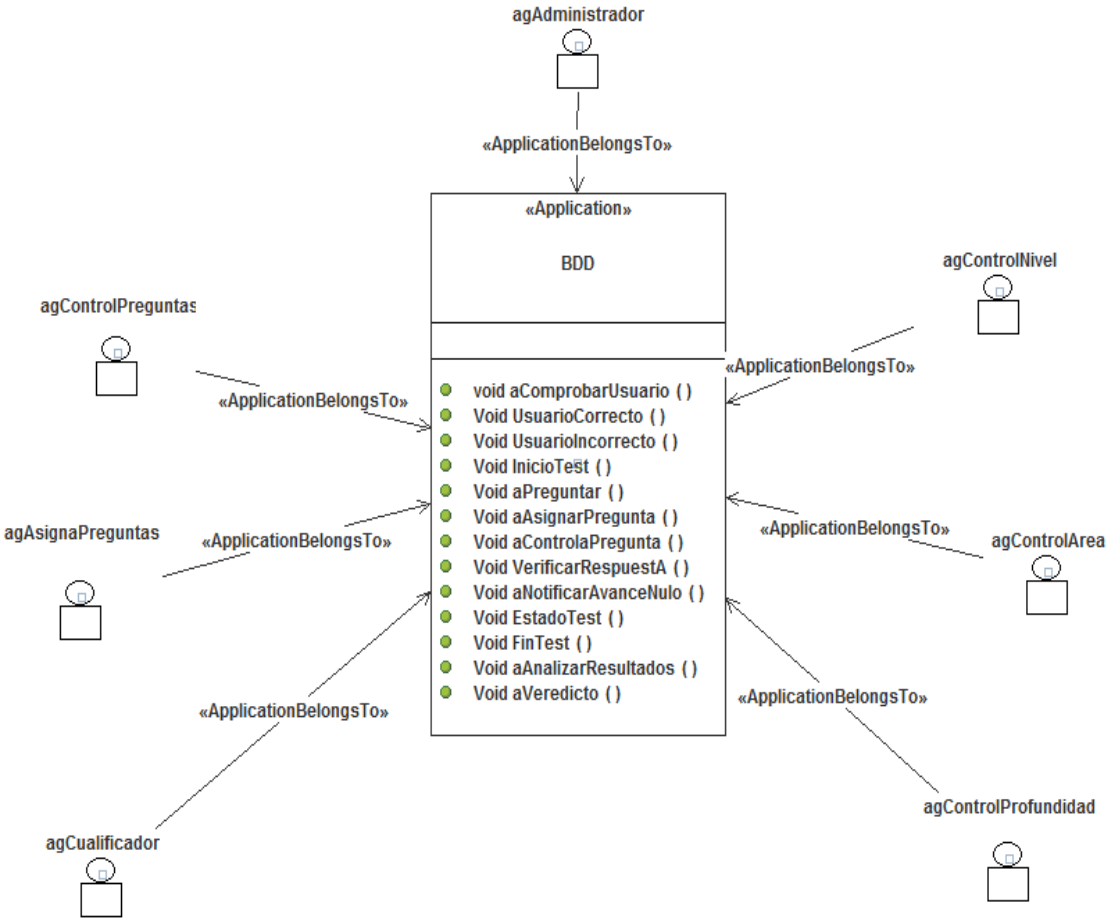


Figura 3.14B Diagrama de Entorno

El diagrama de interacción refleja un esquema de la comunicación entre agentes por medio de mensajes, mismos existentes en el sistema sobre los que se estableció una relación de tipo *UInitiates* con el agente emisor y otra relación de tipo *UColaborates* con el receptor. (Véase figuras 3.15A, 3.15B Diagrama de interacción)

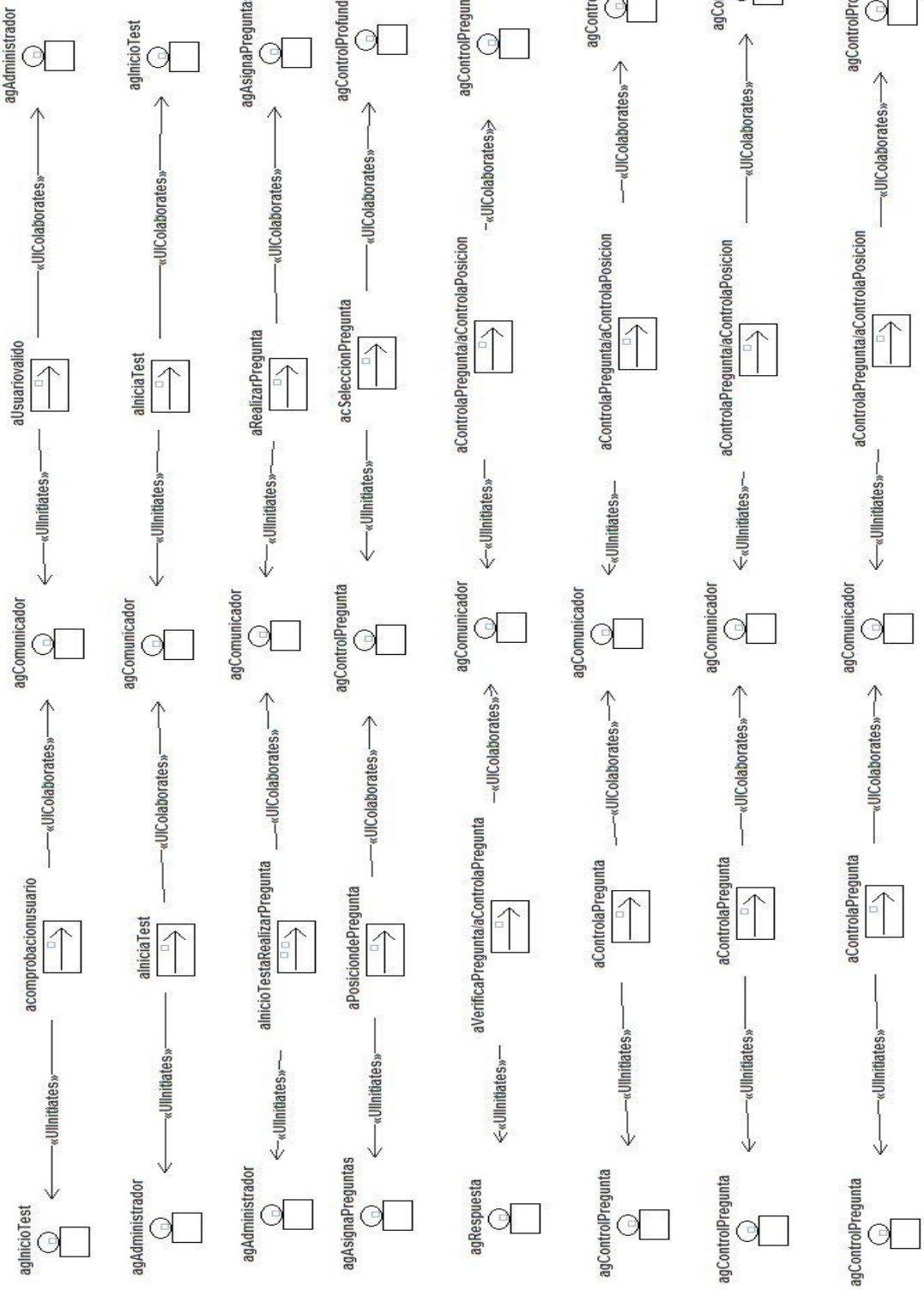


Figura 3.15A Diagrama de Interacción

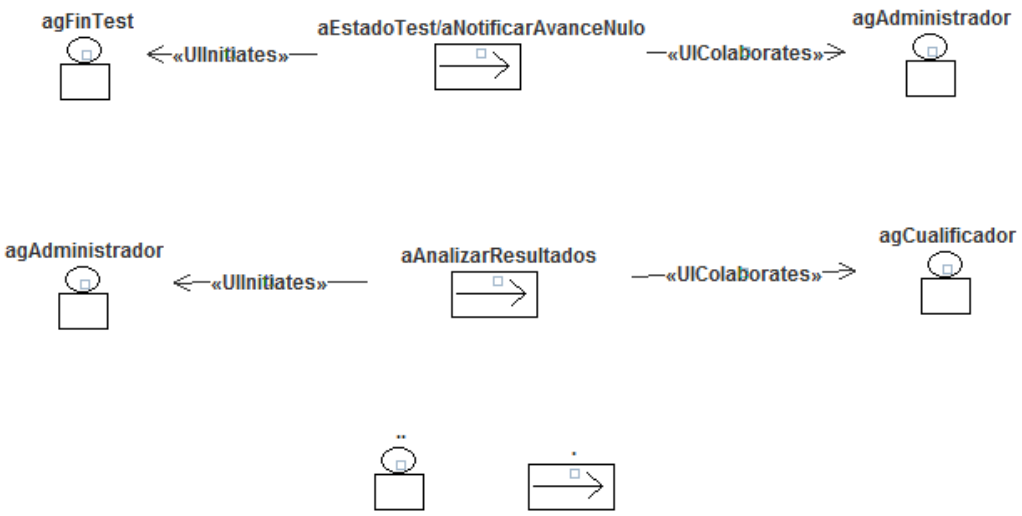


Figura 3.15B Diagrama de interacción

3.5. Identificación de tareas

En el siguiente paso del diseño detallado se identificarán las tareas que ejecutara cada agente. Una tarea, engloba la invocación de uno o varios métodos y/o él envió de mensajes a otros agentes para poder acercarse a sus objetivos como una respuesta a los eventos que recibe de su entorno. Al examinar el modelo de interacción de agentes obtenido, se puede deducir que por cada mensaje recibido por un agente tiene que existir una tarea asociada al agente para atender la llegada del evento. Examinando el modelo inicial obtenido, se puede deducir que por cada percepción que llega a un agente también debe haber una tarea asociada. Se debe aclarar que las tareas se corresponden con las acciones semánticas que ejecutan los agentes.

A continuación, se muestran tablas con las acciones semánticas ejecutadas por cada agente del sistema estos en respuesta a mensajes y percepciones, cada tabla presenta el evento INGENIAS, y su tipo, que provoca la ejecución de la acción semántica. Después la acción semántica ejecutada y una breve descripción.

INGENIAS			
Evento	**Tipo**	**Acción Semántica**	**Descripción**
aComprobarUsuario	AE	Usuario y password obtenidos	Comprueba los datos de acceso al Test mediante el método ***aComprobarUsuario*** del recurso BDD.
aUsuarioCorrecto	IU	Usuario valido	Comunica al agente ***agAdministrador*** que puede iniciar el test.
aUsuarioIncorrecto	IU	Usuario no Valido	No comunica nada al ***agAdministrador*** y se prepara para recibir nuevos datos.

Tabla 3.16 Acciones semánticas ejecutadas por el agente ***agInicioTest.***

En la tabla 3.16 se describen las acciones que tomara el agente *agInicioTest* al iniciar el Test.

INGENIAS			
Evento	**Tipo**	**Acción Semántica**	**Descripción**
mPreguntaSolicitud	P	Solicitud de pregunta	Recibe una notificación del agente ***agInicioTest*** la cual es interpretada como una percepción
MPosicion	M	Consulta de información de posicionamiento	Consulta la posición de la pregunta a asignar al agente ***agControlPregunta***
aAsignarPregunta	AES	Asignación de preguntas al usuario	Asigna las preguntas, estas son adquiridas en la base de datos

Tabla 3.17 Acciones semánticas ejecutadas por el agente ***agAsignaPregunta.***

En la tabla 3.17 se describen las acciones que tomara el agente ***agAsignaPregunta.***

INGENIAS			
Evento	**Tipo**	**Acción Semántica**	**Descripción**
pRespuesta	P	Recepción de respuesta	Recibe la respuesta del usuario la cual es interpretada como una percepción
aVerificaRespuesta	ME	Verificación de respuesta	Consulta en la BDD y verifica la o las respuestas
aRespuestaValidada	M	Envío de respuestas	Envía la validación al agente ***agControlPregunta.***

Tabla 3.18 Acciones semánticas ejecutadas por el agente ***agVerificaRespuesta.***

En la tabla 3.18 se describen las acciones semánticas ejecutadas por el agente ***agVerificaRespuesta,*** este agente se encargara de recibir la respuesta del usuario y verificar las respuestas en la base de respuestas y comunicarlas al agente ***agControlPregunta.***

INGENIAS			
Evento	**Tipo**	**Acción Semántica**	**Descripción**
mPosicion	P	Notificación de posición	Recibe la solicitud de posicionamiento de preguntas a realizar y también recibe la validación de la respuesta esta acción la interpreta como una percepción.
acRegistroPregunta	AC	Guarda respuesta	Registra la respuesta en una matriz de respuestas
aPosicionArea	M	Consulta el área de la pregunta	Comprueba el área de la pregunta realizada al agente ***agControlArea*** y entrega la validación correspondiente.
aPosicionNivel	M	Consulta nivel de la pregunta	Comprueba el nivel de la pregunta realizada al agente ***agControlNivel*** y entrega la validación correspondiente.
aPosicionProfundidad	M	Consulta profundidad de la pregunta	Comprueba la profundidad de la pregunta al agente ***agControlProfundidad*** y entrega la validación correspondiente.
aNotificarAvanceNulo	ME	Notificación de avance nulo	Notifica el avance nulo en el test al agente ***agFinTest*** de existir este caso.

Tabla 3.19 Acciones semánticas ejecutadas por el agente ***agControlaPregunta***

En la tabla 3.19 se describen las acciones semánticas que realizara el agente ***agControlPregunta***, este agente recibirá la posición de las preguntas, registrara las mismas y comunicara la respuesta si fuese necesario, comunicándose con los agentes ***agControlArea***, ***agControlNivel***, ***agControlProfundidad,*** tambien se comunica con el agente ***agFinTest*** de no existir avance en el Test.

INGENIAS			
Evento	**Tipo**	**Acción Semántica**	**Descripción**
aPosicionArea	M	Recibe mensaje de posición	Entrega información de la posición de área en la que se encuentra
mPuntoCriticoNegativo	M	Notificación de cambio de área	Recibe un mensaje del agente ***agControlProfundidad*** de respuesta negativa en un punto critico
aCambioArea	AC	Cambio de área	Realiza el cambio de área en la base de preguntas y deja marcado el nivel en que se encuentra
mTrasladoArea	M	Notificación de cambio de área	Envía un mensaje a los agentes ***agControlNivel*** y ***agControlProfundidad*** el área a trasladarse

Tabla 3.20 Acciones semánticas por el agente ***agControlArea***

En la tabla 3.20 se describen las acciones semánticas realizadas por el agente ***agControlArea***, este es el encargado de comunicar el área a establecerse a los agentes ***agControlNivel*** y ***agControlProfundidad***.

INGENIAS			
Evento	**Tipo**	**Acción Semántica**	**Descripción**
mUltimoNivel	M	Recibe mensaje de posición actual	Recibe un mensaje del agente ***agControlaPregunta*** para entregar información de la posición del nivel en el que se encuentra
mPuntoCritico	M	Notificación de cambio de nivel	Recibe un mensaje del agente ***agControlProfundidad*** de que el agente toma como una percepción
aIncrementaNivel	AC	Incremento de nivel	Realiza el cambio de nivel incrementando el mismo en la matriz de preguntas y envía un mensaje con el número de nivel al agente ***agControlProfundidad*.**
aDecrementaNivel	AC	Decremento de nivel	Realiza el cambio de nivel Decrementando el mismo en la matriz de preguntas y envía un mensaje con el número de nivel al agente ***agControlProfundidad***.
mTrasladoArea	M	Notificación de	Recibe un mensaje del agente

	cambio de área	***agControlArea***, deja marcado la profundidad actual y se traslada al área anunciada.

Tabla 3.21 Acciones semánticas por el agente ***agControlNivel***

En la tabla 3.21 se describen las acciones semánticas ejecutadas por el agente ***agControlNivel,*** este agente recibe el mensaje del agente ***agControlProfundidad*** para decidir si se baja el nivel de las preguntas o se sube el nivel de las mismas.

INGENIAS			
Evento	**Tipo**	**Acción Semántica**	**Descripción**
mUltimoProfundidad	M	Recibe mensaje de ultima ubicación	Recibe un mensaje del agente ***agControlNivel*** para ubicarse en la última pregunta realizada de no existir se ubica en la primera posición.
mPosicionProfundidad	M	Recibe mensaje de posición actual	Recibe un mensaje del agente ***agControlaPregunta*** para entregar información de la posición de la profundidad en la que se encuentra y además recibe la validación de la respuesta
acAvanzaPregunta	AC	Avanza pregunta	El agente decide avanzar una posición en la profundidad de preguntas de ser asertiva la respuesta
mPuntoPriticoPosi	M	Mensaje de punto crítico positivo	El agente se percata de un último punto crítico positivo o última pregunta del nivel y envía el mensaje al agente ***agControlNivel***
acPuntoCriticoNeg	AC	punto crítico negativo	El agente se percata de un punto crítico negativo y avanza de pregunta en profundidad hasta llegar siguiente punto crítico.
mPuntoCriticoNeg	M	Mensaje de punto crítico negativo	El agente se percata de un segundo punto crítico negativo y envía el mensaje al agente ***agControlArea***
acSeleccionPregunta	AC	Aprobación de pregunta	Selecciona la pregunta para ser realizada por el agente
mPosicion	M	Mensaje de confirmación de pregunta	envía el mensaje de confinación de pregunta a realizar al agente ***agControlPregunta***

Tabla 3.22 Acciones semánticas por el agente ***agControlProfundidad***

En la tabla 3.22 se describen las acciones semánticas realizadas por el agente ***agControlProfundidad,*** es este el agente que comunicara a los agentes ***agControlArea***, ***agControlNivel*** la situación actual de las preguntas, estos al ser tomados como estímulos por los agentes se decidirán los cambios a realizarse en el test, también es el encargado de dar luz verde a las preguntas que se realizaran en el test.

INGENIAS			
Evento	**Tipo**	**Acción Semántica**	**Descripción**
mFinTest	M	Mensaje de finalización del test	Recibe el mensaje de finalización del test por el agente ***agFinTest*** este mensaje es tomado como una percepción
pResultado	AES	Recepción y análisis de resultados	Recibe los resultados del agente ***agControlPregunta*** para su análisis.
aVeredicto	AC	Veredicto al test	analiza los resultados con la ayuda de DBB
aEntregaVeredicto	ME	Entrega de resultados	Entrega el veredicto al agente ***agAdministrador*** que se encargara de entregar el mismo al usuario

Tabla 3.23 Acciones semánticas por el agente ***agCualificador.***

En la tabla 3.23 se muestra las acciones que tomara el agente ***agCualificador***, se puede además aprecian su comunicación con otros agentes y la consulta con la base de preguntas multidimensional.

3.6. Especificación del modelo de comportamiento de agentes

El comportamiento de un agente puede describirse mediante un diagrama de estados que indique, para cada estado, los eventos que puede recibir y las acciones semánticas que puede ejecutar como respuesta a cada evento. El autómata resultante se puede obtener a partir de las acciones semánticas expuestas previamente y los eventos que recibe el agente. En la figura 3.15 se muestra, para un agente cualquiera, una transición entre estados. Esta transición se produce cuando el agente recibe un evento (*Evento_Recibido*), para el que

responde con una acción semántica (*Acción_Respuesta_a_Evento*), transitando al estado siguiente (*Estado_Tras_Ejecutar_Acción*).

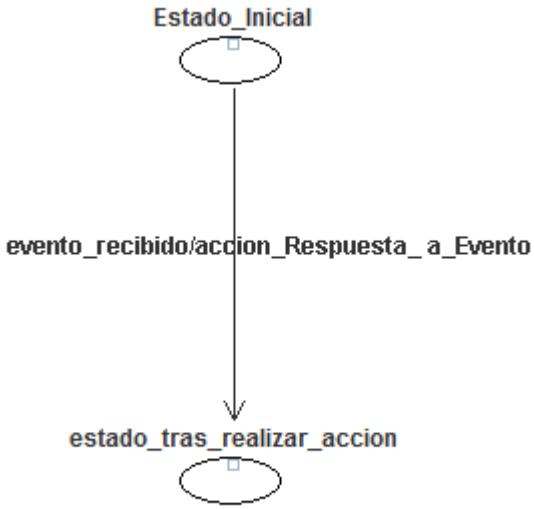


Figura 3.15 Comportamiento de un agente

Existe un estado común en el autómata de todos los agentes que conforman el sistema. Este estado, denominado *UniversalState*, se utiliza para describir la transición universal y representa cualquier estado del autómata. La transición universal se produce cuando un agente, en cualquiera de sus estados, transita al estado ***Fin*** al recibir el evento ***finalizar***. Como respuesta a este evento el agente responde con la acción semántica Nula.

A continuación, se describe el autómata de cada agente obtenido en INGENIAS. Para ilustrar la descripción de los autómatas se incluye una figura del autómata especificado con la herramienta IDK. Además, se muestra la tabla de eventos recibidos que recoge la entidad que emite el evento, el tipo correspondiente en INGENIAS y el contenido del evento.

El autómata del agente ***AgControlProfundidad*** se encuentra inicialmente inactivo (*estadoEsperarMensaje*), una vez que el sistema se inició por completo se percata que los agentes ***agControlArea*** y ***agControlNivel*** le envían un mensaje de ubicación en la que debe

posicionarse y se posiciona en el lugar indicado. Una vez realizada esta acción entra en estado (*estadoEsperarSolicitud*), se percata que recibe la solicitud de ubicación actual y la profundidad de pregunta a realizar, el autómata se percata que no existe validación y realiza la acción semántica (*acSeleccionaPregunta*), entonces realiza la primera pregunta o de recibir la validación de una pregunta de punto crítico avanza al siguiente punto crítico con la acción semántica (*acAvanzaPuntoCritico*), de ser negativa la validación del punto crítico avanza una posición en la profundidad realizando la acción semántica (*acPuntoCriticoNeg*), o de percatarse que es el segundo punto crítico negativo envía un mensaje al agente *agControlNivel*, realizando la acción semántica (*mPuntoCriticoNeg*), y vuelve al estado (*estadoEsperarMensaje*).

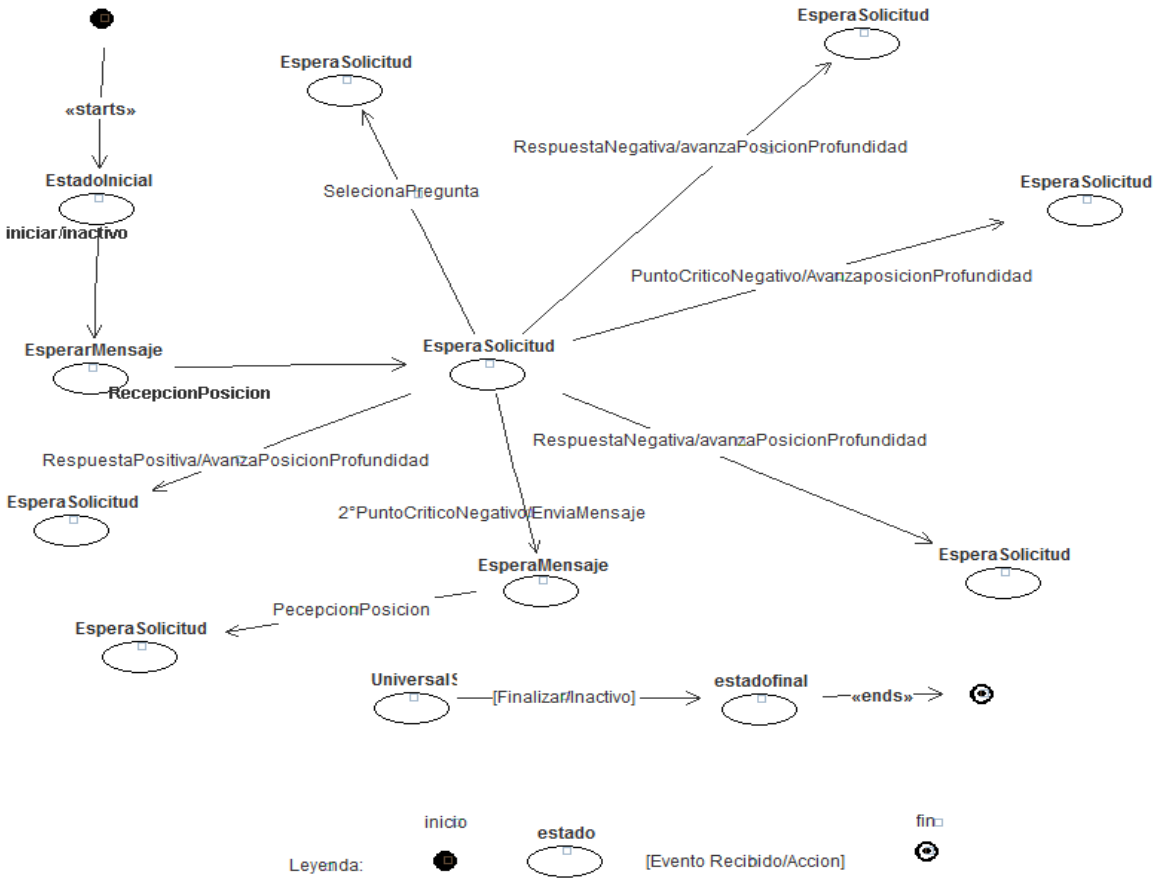


Figura 3.16 Autómata del agente ***agControlProfundidad*** en INGENIAS

El autómata del agente ***agCualificador*** inicialmente se encuentra en espera de que el test finalice (estado *EsperandoMensaje*). La finalización del test se ocasiona cuando no existe avance o cuando se da por satisfecho el proceso de preguntas. Ambos eventos de ven percibidos por el mensaje de finalización del test por el agente ***agFinTest.*** Cuando esto ocurre provoca que el autómata transite al estado (*EsperaResultados)*. El agente ***agControlPregunta*** entrega los resultados y esto conduce al autómata a realizar la acción (*acVeredicto*), en esta parte con los resultados deduce el nivel de conocimiento analizando las preguntas respondidas por nivel y profundidad en cada área, con la ayuda de DBB y guarda la posición de sentencias y las envía al agente ***agAdministrador*** por medio de la acción (*acEntregaVeredicto*), luego de realizada la acción vuelve al estado inicial (*EsperandoMensaje*).

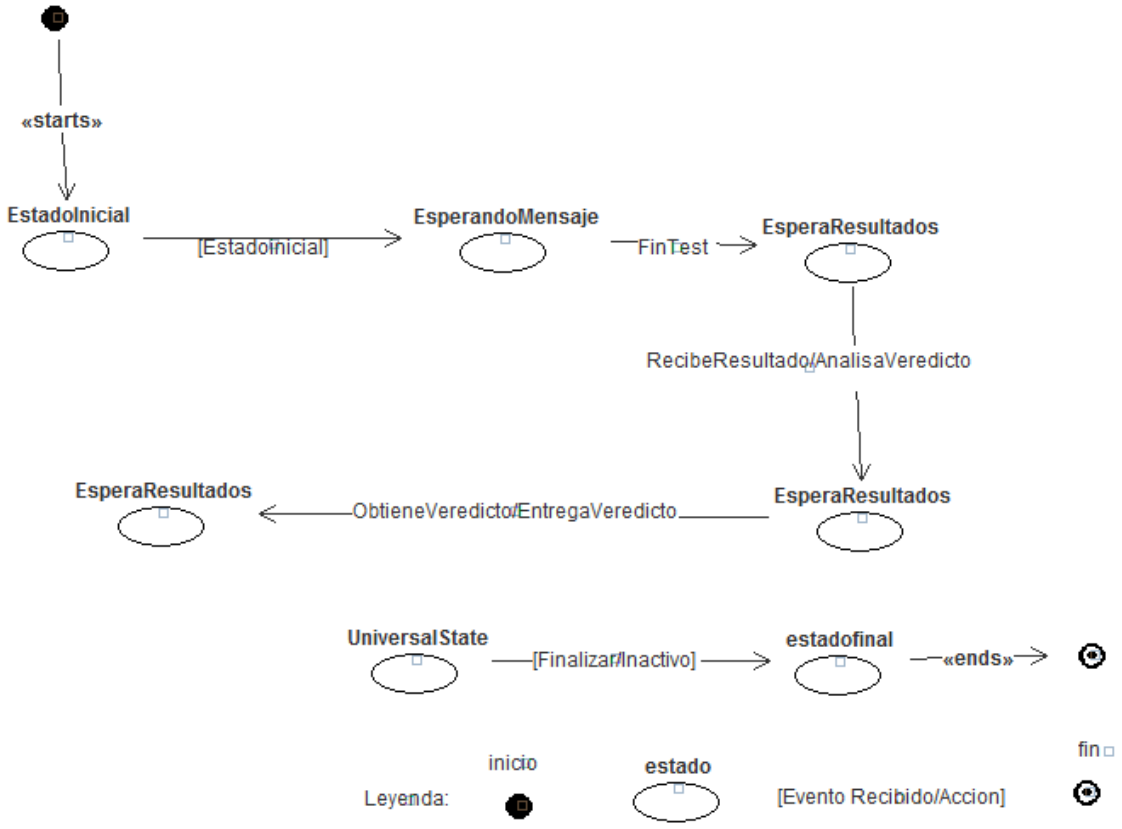


Figura 3.17 Figura 3.16 Autómata del agente ***agControlProfundidad*** en INGENIAS

Lo descrito anteriormente se aplicó en el prototipo y se crearon librerías con el objetivo de emular las acciones de los agentes la aplicación se desarrolló en lenguaje JAVA con MySQL como motor de la base de datos.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package cesartestcuali;

import cesartestcuali.testagent_agAdminitrador;
import cesartestcuali.testagent_agAsignarPregunta;
import cesartestcuali.testagent_agControlPregunta;
import cesartestcuali.testagent_agVerificaRespuesta;
import cesartestcuali.testagent_agControlPregunta;
import cesartestcuali.testagent_agControlNivel;
import cesartestcuali.testagent_agControlArea;
import cesartestcuali.testagent_agControlProfundidad;
import cesartestcuali.testagent_agCualificador;
import cesartestcuali.testagent_agInicioTest;
import cesartestcuali.testagent_agFinTest;
import java.beans.PropertyChangeSupport;
import java.io.Serializable;
import java.util.List;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
```

Figura 3.18 Librerías creadas a partir de los modelos de agente

3.7. Creación de las preguntas del Test

En la parte anterior se describió como los agentes interactuarán en el sistema para poder controlar las diferentes partes de la base de preguntas, en esta parte se detallarán los pasos para la creación de contenidos de esta base de preguntas y como deben ser clasificadas para poder alcanzar la meta de una cualificación confiable.

Uno de los objetivos será extraer los conocimientos de un estudiante tomando en cuenta un mercado laboral en constante evolución como es el informático, por lo tanto, cada pregunta debe evaluar las cualidades, conocimientos y habilidades en el test.

Se sabe que los reactivos de opción múltiple (ROP) o también conocidos como preguntas de reacción múltiple (PRM) son adecuados para medir un amplio espectro de conocimiento y hasta cierto punto la capacidad de aplicarlos, esto dependiendo de que se formulen adecuadamente, además de capacidades como el razonamiento y la toma de decisiones, es de esta manera que se decidirá aferrarse a este tipo de reactivo (pregunta).

Se mencionó en anterior capitulo que el área de evaluación del presente trabajo será el área de ***Redes y Comunicación*** y los contenidos de esta área estarán basados en la propuesta de modelo curricular publicado por la *ACM* y la *sociedad IEEE* en 2016, teniendo en cuenta que *Cisco* es una organización a nivel mundial en provisionar hardware de redes y formación de profesionales en el área ya mencionada, también se tomara como referencia en los reactivos del test.

Se tendrá una serie de reglas a las cuales se regirá la elaboración de cada una de las preguntas a continuación:

- Primeramente, se debe señalar como norma general que los reactivos deben coincidir con los objetivos del área y los temas fundamentales deberán tener más peso que los temas menos importantes, es por este motivo que se debe establecer una tabla de especificaciones para determinar con anterioridad el número de preguntas por cada parte de los temas del área.
- Se debe evitar reactivos que evalúen el conocimiento de datos triviales.
- De ser posible se debe confeccionar preguntas que solo exploren la capacidad de recordar conocimientos abstractos para sacar aprovechar al máximo esta característica que brinda este tipo de reactivos.
- Plantear una misma pregunta en diferentes contextos para poder evaluar si el estudiante sabe aplicar los conocimientos.
- En todos los casos el enunciado de la pregunta debe generar una pregunta clara y debe ser posible llegar a la respuesta con las opciones ocultas.

A continuación, se enuncian algunos de los principales defectos técnicos a evitar:

- No proponer opciones o distractores que no tengan correspondencia gramatical con el enunciado.
- No utilizar opciones globalizadoras (“ninguna de las anteriores” o “todas las respuestas”).
- No repetir elementos en las opciones de tal manera que sean comunes a varias.
- No repetir palabras en el enunciado y en las opciones que puedan facilitar pistas.
- No formulas opciones complejas o dobles.
- No formular enunciados negativos (“cuál no es”, “cuál es falsa”, “todas son ciertas excepto…”).

Se muestra en las Figuras 3.19 - 3.21 algunas de las preguntas elaboradas a partir de las indicaciones descritas anteriormente que serán usadas en el test.

Contexto: Este tipo de pregunta se desarrolla en torno a un (1) enunciado y cuatro (4) opciones de respuesta (A, B, C, D). Solo una (1) de estas opciones responde correctamente a la pregunta

Enunciado: Leonardo es el administrador de una LAN Ethernet que consta de 10 estaciones de trabajo, 2 servidores y 3 impresoras, todo ello conectado a través de un hub central de 20 puertos. El rendimiento de la red ha disminuido bastante como consecuencia de la instalación de una nueva aplicación, crítica para la Organización, que genera muchas colisiones de paquetes, haciendo lento los accesos a datos y servicios. Los usuarios presionan al Director de Sistemas de Información para que se solucione el problema. El Director pide a Leonardo su opinión sobre la solución más económica y eficiente. El consejo de Leonardo debería ser:

Seleccione una respuesta.

a. Sustituir el cableado UTP por fibra óptica para aumentar el ancho de banda.
b. Sustituir los dos servidores actuales por un servidor tetraprocesador
c. Desinstalar la aplicación que causa los problemas.
d. Sustituir el hub por un switch (Respuesta)

Figura 3.19. Ejemplo de pregunta (*Redes de área local NC*)

En la Figura 3.19 se puede apreciar que la pregunta fue formulada en referencia a una situación real para evaluar la parte aplicativa del tema abordado.

Contexto: Este tipo de pregunta se desarrolla en torno a un (1) enunciado y cuatro (4) opciones de respuesta (A, B, C, D). Solo una (1) de estas opciones responde correctamente a la pregunta

Enunciado: TIA/EIA-568-B intenta definir estándares que permitirán el diseño e implementación de sistemas de cableado estructurado para edificios comerciales y entre edificios en entornos de campus. La forma de organizar los diferentes pares de hilos del cable UTP según el estándar 568-B es:

Seleccione una respuesta.

a. Blanco Verde/Verde, Blanco Naranja/Azul, Blanco Azul/Naranja, Blanco Café/Café.
b. Blanco Naranja/Azul, Blanco Azul/Naranja, Blanco Verde/Verde, Blanco Café/Café.
c. Blanco Naranja/Naranja, Blanco Verde/Azul, Blanco Azul/Verde, Blanco Café/Café (Respuesta)
d. Blanco Azul/Naranja, Blanco Naranja/Azul, Blanco Verde/Verde, Blanco Café/Café.

Figura 3.20. Ejemplo de pregunta (*redes de área local NC*)

En la figura 3.20. Se puede apreciar conocimientos de estándares esta pregunta es de manera directa y cumple con las características ya mencionadas.

Contexto: Este tipo de pregunta se desarrolla en torno a un (1) enunciado y cuatro (4) opciones de respuesta (A, B, C, D). Solo una (1) de estas opciones responde correctamente a la pregunta

Enunciado: Son los dispositivos de regulación de tráfico más importantes en las redes grandes. Tienen acceso a las direcciones del nivel de red y contienen software que permite determinar cuál de los posibles caminos entre esas direcciones es el mejor para la transmisión determinada. Permiten que prácticamente cualquier tipo de ordenador se pueda comunicar con otro en cualquier parte del mundo.

Seleccione una respuesta.

a. Puentes.
b. Hub
c. Routers (Respuesta)
d. Bridge o Puente

Figura 3.21. Ejemplo de pregunta (*Enrutamiento y Reenvio NC*)

Contexto: Este tipo de preguntas consta de dos proposiciones, así: una Afirmación y una Razón, Unidas por la palabra PORQUE. El estudiante debe examinar la veracidad de cada proposición y la relación teórica que las une. Para responder este tipo de preguntas se debe leer toda la pregunta y señalar la respuesta elegida de acuerdo con las siguientes instrucciones:

Marque A si la afirmación y la razón son VERDADERAS y la razón es una explicación CORRECTA de la afirmación.

Marque B si la afirmación y la razón son VERDADERAS, pero la razón NO es una explicación CORRECTA de la afirmación.

Marque C si la afirmación es VERDADERA, pero la razón es una proposición FALSA.

Marque D si la afirmación es FALSA, pero la razón es una proposición VERDADERA.

Enunciado: El mecanismo de acceso al medio usado en una Ethernet se denomina portadora sensible a acceso múltiple con detección de colisiones (CSMA/CD), carrier Sense Múltiple Access UIT Collision Detection) (estandarizado en el IEEE 802.3). PORQUE Siempre que múltiples usuarios tienen acceso incontrolado a una única línea, existe el peligro de que las señales se solapen y se destruyan entre sí. Estos solapamientos, que convierten las señales en ruido inútil, se denominan colisiones. A medida que se incrementan el tráfico en un enlace con múltiples accesos, se incrementan las colisiones.

Seleccione una respuesta.

a. Marque A si la afirmación y la razón son VERDADERAS y la razón es una explicación CORRECTA de la afirmación. (Respuesta)
b. Marque B si la afirmación y la razón son VERDADERAS, pero la razón NO es una explicación CORRECTA de la afirmación.
c. Marque C si la afirmación es VERDADERA, pero la razón es una proposición FALSA.
d. Marque D si la afirmación es FALSA, pero la razón es una proposición VERDADERA.

Figura 3.22. Ejemplo de pregunta (*Movilidad NC*)

En la figura 3.22 anterior se tiene una pregunta de conocimiento, abstracción y aplicación respectivamente. Cabe recalcar que este tipo de preguntas deben ser ubicados en la matriz de preguntas de manera ascendente en dificultad tanto en los niveles y la profundidad.

CAPÍTULO IV

PRUEBA DE HIPÓTESIS

4.1. Introducción

En este capítulo se desplegará la solución estadística aplicada a la herramienta (test administrado por un Sistema Multi-Agente) para la cualificación de conocimientos en estudiantes de la carrera de informática, para la comprobación de la hipótesis, en este caso se utilizó la prueba no paramétrica aleatoria de Rachas, esto para comprobar que la muestra sometida al sistema multi-agente es aleatoria, después se comprobara el porcentaje de aceptación al veredicto del sistema multi-agente en el test que debe ser al menos de un 70% en la población.

A continuación, se detalla el método a ser usado para la demostración, seguidamente se aplicará las formulas a los datos obtenidos en el sistema multi-agente.

4.2. Prueba de rachas de Wald-Wolfowitz

La Prueba de rachas es utilizada para determinar si una muestra de observaciones es o no aleatoria, es decir, para determinar si las observaciones de una determinada secuencia son independientes entre sí, En una serie temporal, por ejemplo, las observaciones no son aleatorias: lo que ocurre con una observación cualquiera depende generalmente de las características de la observación anterior. En una muestra aleatoria, por el contrario, debemos esperar que lo que ocurre con una observación cualquiera sea independiente de las características (y de la siguiente).

El concepto de Racha hace referencia a una secuencia de observaciones de un mismo tipo. Supongamos que lanzamos una moneda al aire 10 veces y que obtenemos el siguiente resultado:

CCCXCCXXXC

En el ejemplo tenemos 5 rachas: CCC, X, CC, XXX, C. A simple vista el resultado obtenido parece aleatorio. Pero si en lugar de ese resultado hubiéramos obtenido este otro:

CCCCCXXXXX se tiene (2 rachas)

Resultaría fácil ponerse de acuerdo en que la secuencia obtenida no parece aleatoria. Como tampoco parece aleatoria una secuencia con demasiadas rachas:

CXCXCXCXCX (10 rachas)

Una vez visto esto, la prueba de las rechas permite determinar si el número de rachas (R) observado en una determinada muestra de tamaño n es lo suficientemente grande o lo suficientemente pequeño como para poder rechazar la hipótesis de independencia o aleatoriedad entre las observaciones.

Para obtener el número de rachas es necesario que las observaciones estén clasificadas en dos grupos exhaustivos y mutuamente exclusivos.

Una vez clasificadas las ***n*** observaciones en dos grupos (de tamaños n_1 y n_2), se utiliza una tipificacion[3] del número de rachas denotado por (R) para contrastar la hipótesis de aleatoriedad o independencia:

$$Z = \frac{R - E(R)}{\sigma_R}$$

El termino estadístico $\boldsymbol{Z}$ se distribuye según el modelo de probabilidad normal $N(0,1)$. Para muestras aleatorias, la distribución de probabilidad de R tiende hacia la normal, a medida que n_1 y n_2, se van agrandando, de tal manera que:

$$\boldsymbol{R \rightarrow N(E[R]), \sqrt{Var[R]}}$$

Esperanza	$$E(R) = \frac{(2 * n_1 * n_2)}{n + n_2}$$
Varianza	$$Var(R) = \frac{(2n_1n_2(2n_1n_2 - n_1 - n_2))}{(n_1 + n_2)^2((n_1 + n_2) - 1)}$$

Tabla 4.1. Esperanza y Varianza

Por consiguiente, para muestras grandes se verifica:

$$Z = \frac{R - E(R)}{\sqrt{Var(R)}}$$

Y para una muestra concreta el valor estadístico Z será:

$$Z_{exp} = \frac{R - \left(\frac{2n_1 n}{n} + 1\right)}{\sqrt{\frac{2n_1n_2(2n_1n_2 - n)}{n^2(n-1)}}}$$

En donde R es el número total de rachas observadas en la muestra.

La región de aceptación para la hipótesis nula será:

$$-Z_{\frac{\alpha}{2}} < Z_{exp} < Z_{\frac{\alpha}{2}}$$

El valor de $-Z_{\frac{\alpha}{2}}$ se obtiene de la tabla de la N (0.1) de manera que:

$$P\left(Z_1 < -Z_{\frac{\alpha}{2}}\right) = p\left(Z_1 < Z_{\frac{\alpha}{2}}\right) = \frac{\alpha}{2}$$

4.3. Desarrollo de la prueba de hipótesis

Se puede apreciar que la aceptación de los estudiantes es

Para el desarrollo de la prueba de hipótesis por medio del contraste de rachas de Wald-Wolfowitz se sigue los siguientes pasos:

Paso 1. Planteamiento de la hipótesis nula.

El uso de un sistema multi-agente para la cualificación de conocimientos en estudiantes egresados de la carrera de Informática determina un rechazo al resultado de manera negativa en menos de un 30%.

Paso 2. Nivel de significación

Para una muestra de 22 personas el nivel de significancia o confianza elegida de la Tabla Normal es $\alpha = 0.05$.

Paso 3. Identificación estadística de prueba

Para este caso se utiliza la prueba de rachas de Wald-Wolfowitz utiliza los signos de los residuos y sus variaciones de negativos y positivos o viceversa. Una racha vendrá constituida por la sucesión de signos iguales.

Paso 4. Formulación de la regla de decisión.

Para la prueba, se tomó como muestra a 19 personas que son estudiantes egresados de la carrera de informática y estudiantes de 8^{vo} semestre, puesto que el tema abarca el área de Redes de Comunicación y dicha materia se dicta a estudiantes de 7^{mo} semestre, estos estudiantes probaron el sistema sometiéndose al test, al recibir el veredicto ellos dieron su aceptación o su rechazo al mismo, este valor se llevó a la siguiente tabla.

Nº	Aceptación al veredicto del test	Rechazo al veredicto del test	Aceptación por rachas
1.	X		+
2.	X		+
3.	X		+
4.		X	-
5.		X	-
6.	X		+
7.	X		+
8.	X		+
9.	X		+
10.		X	-
11.	X		+
12.	X		+
13.	X		+
14.		X	-
15.	X		+
16.		X	-
17.	X		+
18.	X		+
19.	X		+

Tabla 4.2. Aceptación o rechazo al veredicto al test de cualificación

Se obtienen los resultados:

(+ + +)(- -)(+ + + +)(-)(+ + +)(-)(+)(-)(+ + +)

Donde:

- (-) Representa el rechazo al veredicto del test para cualificación administrado por un agente inteligente.
- (+) Representa la aceptación al veredicto del test para cualificación administrado por un agente inteligente.

Siendo una racha construida por la sucesión de signos iguales se tiene:

Total de rachas expuestas	$\boldsymbol{R_{exp}}$ **= 9**
Número total de observaciones	N = 19
Numero de residuos positivos	n_1=14
Numero de residuos negativos	n_2=5

Reemplazando los datos para calcular la *Esperanza* y *Varianza* se tiene:

Esperanza	$\boldsymbol{E[R] = \frac{2n_1n_2}{n} + 1 = \frac{2 * 14 * 5}{14 + 5} + 1 = \frac{140}{19} + 1 = 8.36}$
Varianza	$Var[R] = \frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2((n_1 + n_2) - 1)}$ $= \frac{2 * 14 * 5 * (2 * 14 * 5 - 14 - 5)}{(14 + 5)^2((14 + 5) - 1)} = \frac{16940}{6498} = 2.61$

Paso 5. Toma de decisión

Para una muestra concreta el valor estadístico Z_{exp} reemplazando datos se tiene:

$$Z_{exp} = \frac{R - E(R)}{\sqrt{Var(R)}} = \frac{9 - 8.36}{\sqrt{2.61}} = \frac{0.64}{1.6155} = 0.3961$$

Para calcular la región de aceptación de la hipótesis es necesario hallar el valor de $Z_{\frac{\alpha}{2}}$ que se obtiene de la tabla N(0,1), de manera que cumple:

$$P\left(Z_1 \leq -Z_{\frac{\alpha}{2}}\right) = p\left(Z_1 \geq Z_{\frac{\alpha}{2}}\right) = \frac{\alpha}{2}$$

$$P\left(Z_1 \leq -Z_{\frac{\alpha}{2}}\right) = \frac{\alpha}{2} \qquad P\left(Z_1 \geq Z_{\frac{\alpha}{2}}\right) = \frac{\alpha}{2}$$

$$1 - P\left(Z_1 < Z_{\frac{\alpha}{2}}\right) = \frac{\alpha}{2} \qquad \left(Z_1 \geq Z_{\frac{\alpha}{2}}\right) = 0.025$$

$$P\left(Z_1 < Z_{\frac{\alpha}{2}}\right) = 1 - \frac{0.05}{2} \qquad Z_{\frac{\alpha}{2}} = 1.96$$

$$= 0.975$$

$$Z_{\frac{\alpha}{2}} = -1.96$$

Como regla de decisión al 90% de confianza, no se rechazara la hipótesis nula de aleatoriedad H_0 si Z_{exp} se encuentra en el intervalo:

$$-Z_{\frac{\alpha}{2}} < Z_{exp} < Z_{\frac{\alpha}{2}}$$

$$-1.96 < 0.3961 < 1.96$$

Se puede ver que el valor estadístico $Z_{exp} = 0.3961$ se encuentra en el intervalo de la hipótesis, por lo tanto se puede afirmar:

"El uso de un sistema multi-agente para la cualificación de conocimientos en estudiantes egresados de la carrera de Informática determina un rechazo al resultado de manera negativa en menos de un 30%."

Sabiendo que la muestra es aleatoria procedemos al cálculo de la aceptación al veredicto brindado por el sistema multi-agente.

Haciendo uso de una simple regla de tres tenemos

N=19: número total de estudiantes sometidos al test
n=14 número de estudiantes que aceptaron el veredicto del test

$$x = \frac{n}{N} * 100 = \frac{14}{19} * 100 = 73.68$$

Por el complemento tenemos 26,32% de rechazo al resultado brindado por el test.

Por lo tanto, sabiendo que el porcentaje de aceptación al veredicto del sistema multi-agente en el test es de 73.68%, que demuestra que la tesis es un trabajo valido y además revela que los datos de la muestra para la aceptación al veredicto son aleatorios.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

El modelado de los agentes en un SMA para la cualificación de conocimientos haciendo uso de un test de opción múltiple en estudiantes de la carrera de informática presenta una importante contribución no solo a la línea de investigación educacional o de sistemas multi-agente sino en el proceso de mejorar las habilidades y conocimientos de los estudiantes, puesto que los parámetros que recomienda el presente trabajo hace referencia a organizaciones reconocidas no solo en nuestro medio sino a nivel internacional la cual lleva décadas a la vanguardia en el área de las ciencias de la computación.

- Se aplicaron niveles de preguntas los cuales ayudaron a determinar el nivel de conocimiento de los estudiantes egresados sometidos al test
- Se aplicaron métricas para la elaboración de preguntas basadas en el modelo curricular 2016 publicado por la ACM & IEEE, también basado en el curso de redes CISCO (CCNA).
- Se aplicaron reglas de elaboración de preguntas haciendo uso de reactivos de opción múltiple.
- Se analizaron los casos de interacción del sistema con el usuario tomando en cuenta las respuestas como percepciones de los agentes
- Los parámetros aplicados a las preguntas del test poseen un contexto de situaciones reales de trabajo
- Se hizo uso de la unificación de metodologías como ser: Prometheus e INGENIAS para el modelado del sistema.

  Se comprueba una vez más que la unión de metodologías de sistemas multi-agente que complementen y enriquezcan el modelado de agentes inteligentes y su interacción

dentro de un sistema es viable y sostenible puesto que se obtiene una mejor visión de los modelos AUML y también de la Ingeniería de software orientado a agentes.

5.2. Recomendaciones

El modélalo y el prototipo descrito en el presente trabajo aborda el análisis y el diseño de un test administrado por un agente que a su vez cuenta con agentes colaboradores para lograr cualificar conocimientos apoyado en la correcta elaboración de las preguntas que conforman el test.

Se presentan las siguientes recomendaciones:

- Se recomienda el uso de Jade como plataforma de desarrollo para los agentes puesto que el modelado en INGENIAS es una herramienta que pertenece al lenguaje Java igual que Jade.
- Se recomienda quitar los acotamientos del presente trabajo y ampliar la presente investigación puesto que el tema es de cobertura amplia y exquisita.
- Recomienda profundizar en la iteración y roles de los agentes para un mejor desempeño incluso en casos atípicos de un SMA.
- Se recomienda conformar un equipo de especialistas para poder potenciar las preguntas del test, por recomendaciones de un experto en el área educacional como ser el PH.D. Franz Gutiérrez profesor venezolano hacer el uso de reactivos de respuesta al ítem para una mejor extracción de información en el estudiante.

BIBLIOGRAFÍA

Association for Computing Machinery (ACM), I. C. (December 20,2016). *Computer Science Curricula 2016.* United States of America: Robert Vizzini.

Brown. S (2007). Evaluar en la universidad: problemas y nuevos enfoques. Madrid Narcea ediciones.

Choque. (2013). Agentes inteligentes.

Fernández, Y., & Díaz, Y. (2012). *Patrón Modelo-Vista-Controlador.* Obtenido de http://revistatelematica.cujae.edu.cu/index.php/tele/article/viewFile/15/10

Gómez, Jorge & Pavón, Juan (2005) INGENIAS Development Kit (IDK) Manual. Madrid España, Universidad Compútense de Madrid.

Gomez, C. (2000) Del etiquetado de las ocupaciones según nivel de cualificación. Sociologia del Trabajo N 39, p.33-61

Hanckes, H. (s.f.). *Tutorial 10 - Modelo Vista Controlador.* Pontificia Universidad Católica de Chile. Escuela de Ingeniería. Departamento de Ciencia de la Computación. Obtenido de http://revistatelematica.cujae.edu.cu/index.php/tele/article/viewFile/15/10

L., H. R. (2006). *Metodologia de la Investigacion.* Sampieri.

Ling Padgham, J. T. (s.f.). *Prometheus Design Tool version 2.5.* Obtenido de RMIT Agent Group: http://www.es.rmit.edu.eu/agents/pdt/

Mestras, J. G. (s.f.). *Desarrollo de Sistemas Multi-Agente con INGENIAS.* Universidad Complutense de Madrid , Departamento de Sistemas Informaticos y Programacion.

palfrei. (30 de mayo de 2015). *mozilla*. Recuperado el 24 de noviembre de 2015, de https://developer.mozilla.org

Proyecto Fin de Máster, Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid.

Sanz., J. G. (2002). *Modelado de Sistemas Multiagente*. Universidad Complutence de Madrid. Madrid: Departamento de Sistemas Informaticos.

Rigby & Sanchis (2006) El concepto de cualificación y su construcción social. Revista Europea de Formación Profesional N 37 - 2006/1

Winikoff, L. P. (2004). *Developing Intelligent Agent Syatems.* John Wiley and Sons.

Rodríguez, A. (2009) Asistente para la Gestión de Organizaciones de Agentes Software.

Romero, J.; Dafonte, C.; Gomez, A.; Penousal, F. 2007 Inteligencia Artificial y Computación Avanzada

ANEXOS.

1. Una empresa está contemplando la posibilidad de utilizar una red cliente / servidor o una red peer-to-peer. ¿Cuáles son las tres características de una red peer-to-peer? (Elige tres.)

mejor seguridad

Fácil de crear *

Mejor rendimiento del dispositivo cuando actúa como cliente y servidor

Carece de administración centralizada *

Menos costo de implementar *

Escalable

Preguntas de nivel 1 usadas en el test

2. ¿Qué dispositivo realiza la función de determinar la ruta que los mensajes deben tomar a través de redes internas?

Un enrutador

Un firewall

Un servidor web

Un modem DSL

Preguntas de nivel 1 usadas en el test

3. ¿Qué dos criterios se utilizan para ayudar a seleccionar un medio de red de varios medios de red? (Escoge dos.)

Los tipos de datos que deben priorizarse

El coste de los dispositivos finales utilizados en la red

La distancia que el medio seleccionado puede llevar con éxito una señal *

El número de dispositivos intermedios instalados en la red

El entorno en el que se va a instalar el medio seleccionado *

Preguntas de nivel 1 usadas en el test

4. ¿Cuáles dos declaraciones describen dispositivos intermedios? (Escoge dos.)

Los dispositivos intermedios generan contenido de datos.

Los dispositivos intermedios alteran el contenido de los datos.

Los dispositivos intermediarios dirigen la trayectoria de los datos. *

Los dispositivos intermedios conectan hosts individuales a la red. *

Los dispositivos intermedios inician el proceso de encapsulación.

Pregunta de nivel 1 usada en el test

5. ¿Cuáles son las dos funciones de los dispositivos finales en una red? (Escoge dos.)

Originan los datos que fluyen a través de la red. *

Direccionan datos sobre rutas alternativas en caso de fallos de enlaces.

Filtran el flujo de datos para mejorar la seguridad.

Son la interfaz entre los seres humanos y la red de comunicación. *

Proporcionan el canal sobre el cual el mensaje de la red viaja.

Pregunta de nivel 1 usada en el test

6. ¿Qué área de la red sería más probable que un personal universitario de TI tenga que rediseñar como resultado directo de que muchos estudiantes traigan sus propios tabletas y teléfonos inteligentes a la escuela para acceder a los recursos de la escuela?

Extranet

Intranet

LAN por cable

LAN inalámbrico*

WAN inalámbrica

Pregunta de nivel 1 usada en el test

7. ¿A qué tipo de red debe acceder un usuario doméstico para hacer compras en línea?

Una intranet

La Internet*

Una extranet

Una red de área local

Pregunta de nivel 1 usada en el test

8. Un empleado de una sucursal está creando una cotización para un cliente. Para ello, el empleado debe tener acceso a la información confidencial de precios de los servidores internos de la oficina central. ¿A qué tipo de red accedería el empleado?

Una intranet

La Internet

Una extranet

Una red de área local

Pregunta de nivel 1 usada en el test

9. ¿Cuáles dos opciones de conexión proporcionan una conexión a Internet de alto ancho de banda siempre a los ordenadores en una oficina en casa? (Escoge dos.)

celular

DSL *

satélite

cable*

Teléfono de marcación

Pregunta de nivel 1 usada en el test

1. ¿Cuál es la función del kernel de un software operativo?

Proporciona una interfaz de usuario que permite a los usuarios solicitar una tarea específica.

El kernel enlaza los controladores de hardware con la electrónica subyacente de una computadora.

Es una aplicación que permite la configuración inicial de un dispositivo Cisco.

El kernel proporciona recursos de hardware para cumplir con los requisitos de software. *

Pregunta de nivel 2 usada en el test

2. Un administrador de red debe mantener el ID de usuario, la contraseña y el contenido de la sesión privados al establecer la conectividad CLI remota con un conmutador para gestionarla. ¿Qué método de acceso debe elegirse?

Telnet

Consola

AUX

SSH *

Pregunta de nivel 2 usada en el test

6. ¿Cuáles son las dos funciones que se proporcionan a los usuarios mediante la función de ayuda contextual de la CLI de Cisco IOS? (Escoge dos.)

Proporcionando un mensaje de error cuando se envía un comando incorrecto

Mostrando una lista de todos los comandos disponibles dentro del modo actual *

Permitiendo al usuario completar el resto de un comando abreviado con la tecla TAB

Determinar qué opción, palabra clave o argumento está disponible para el comando introducido *

Seleccionar el mejor comando para realizar una tarea

Pregunta de nivel 2 usada en el test

2. ¿Cuáles tres capas del modelo OSI son comparables en función de la capa de aplicación del modelo TCP / IP? (Elige tres.)

solicitud*

presentación*

sesión*

transporte

enlace de datos

físico

red

Pregunta de nivel 3 usada en el test

3. Un administrador de red advierte que algunos cables Ethernet recién instalados llevan señales de datos corruptas y distorsionadas. El nuevo cableado se instaló en el techo cerca de luces fluorescentes y equipos eléctricos. ¿Cuáles dos factores pueden interferir con el cableado de cobre y resultar en distorsión de señal y corrupción de datos? (Escoge dos.)

EMI *

Diafonía

RFI *

Atenuación de la señal

Longitud de cableado

Pregunta de nivel 3 usada en el test

18. ¿Cuáles son dos posibles problemas de red que pueden resultar del funcionamiento del ARP? (Escoge dos.)

La configuración manual de las asociaciones ARP estáticas podría facilitar el envenenamiento por ARP o la suplantación de direcciones MAC.

En redes grandes con poco ancho de banda, múltiples transmisiones ARP podrían causar retrasos en la comunicación de datos. *

Los atacantes de red podrían manipular direcciones MAC y asignaciones de direcciones IP en mensajes ARP con la intención de interceptar el tráfico de red. *

Un gran número de emisiones de solicitud de ARP podría hacer que la tabla de direcciones MAC del host se desborde y evitar que el host se comunique en la red.

Varias respuestas ARP resultan en la tabla de direcciones MAC del conmutador que contiene entradas que coinciden con las direcciones MAC de los hosts que están conectados al puerto de conmutador relevante.

Pregunta de nivel 3 usada en el test