Vol. III, No. 3, septiembre - diciembre, 2012, pp. 47 - 56

Propuesta de actividades de pruebas para Ingenias

Yahima Hadfeg Fernández
Correo electrónico:yhadfeg@ceis.cujae.edu.cu

Mailyn Moreno Espino
Correo electrónico:my@ceis.cujae.edu.cu

Martha Dunia Delgado Dapena
Correo electrónico:marta@ceis.cujae.edu.cu
Instituto Superior Politécnico José Antonio Echeverría, La Habana, Cuba

Artículo de Reflexión

Resumen
En este trabajo se recopila el estado en que se encuentra la fase de pruebas en los agentes y sistemas multiagentes utilizando diferentes metodologías. En el desarrollo del mismo se explican los tipos de pruebas que se les deben realizar a los sistemas multiagentes, así como las diferentes metodologías orientadas a agentes para cubrir esta importante fase. Se exponen características esenciales de las herramientas que soportan el diseño de estas metodologías; además de los módulos o mecanismos que se les han incorporado para abarcar esta problemática que impacta directamente en la calidad del software. Ingenias es una de las metodologías orientadas a agentes que carece de una etapa de prueba funcional. Aquí se proponen pruebas de aceptación y pruebas de sistema para esta metodología, tomando como base V-Model y permitiendo la construcción de un software con mayor calidad.

Palabras clave: agentes, pruebas en agentes, Ingenias, V-Model, prueba de aceptación, prueba de sistema

Recibido: 23 de mayo del 2012 Aprobado: 5 de julio del 2012

INTRODUCCIÓN

La fortaleza y calidad del software son aspectos muy importantes, aunque en ocasiones no se tienen en cuenta. [1] Propiciar la calidad en el software es una actividad que ha surgido como consecuencia de la fuerte demanda de sistemas de software en todos los procesos que se desarrollan en la actualidad; desde programas elementales de contabilidad hasta programas complejos como los espaciales. De ahí el esfuerzo que se ha desplegado para obtener software de alta calidad. La ingeniaría de software aporta herramientas y procedimientos para garantizar la fortaleza y calidad requeridas.

Las pruebas son generalmente consideradas costosas y molestas, pero es una molestia necesaria; una buena práctica es probar en etapas tempranas y frecuentes. Mientras más se pruebe el software, mayor cantidad de errores serán encontrados (aunque las malas estrategias de prueba son a menudo las más usadas y las más ineficaces).

Los agentes inteligentes son un nuevo paradigma que surge tras la necesidad de resolver problemas que con la orientación a objetos (OO) no se logran solucionar o la solución es más compleja e ineficiente. Debido al comportamiento emergente de los sistemas multiagentes (SMA) y las características que ellos poseen es difícil realizar pruebas tal y como se realizan en el paradigma OO. [2]

La mayoría de las metodologías orientadas a agentes no cuentan con una fase de prueba bien estructurada, por ser tan recientes y por las características de los agentes. [3] Al ser las pruebas tan necesarias, existen trabajos para comenzar a incorporar algunas actividades de pruebas en dichas metodologías.

Este trabajo se centra en realizar un estudio del estado del arte de las pruebas por cada metodología, con el objetivo de ilustrar el avance de cada una de las metodologías en este sentido y proponer actividades de prueba a la metodología Ingenias.

AGENTES Y SISTEMAS MULTIAGENTES

Los agentes inteligentes (AI) son programas, software con características peculiares que se diferencian de los software convencionales. Existe una diversidad de conceptos acerca de los agentes, aunque hay un consenso de que un agente es una entidad computacional que es capaz de actuar de manera autónoma dentro de ambientes dinámicos y abiertos. [4,5]

Por lo general, los agentes no son desarrollados solos, sino como entidades que constituyen un sistema. A dicho sistema se le denomina multiagente. En este caso, los agentes deben o pueden interactuar entre ellos. Las interacciones más habituales como son informar o consultar a otros agentes permiten a los agentes "hablar" entre ellos, tener en cuenta lo que realiza cada uno de ellos y razonar acerca del papel desempeñado por los diferentes agentes que constituyen el sistema. La comunicación entre agentes se realiza por medio de un lenguaje de comunicación de agentes (uno de los más conocidos es FIPA-ACL).

Los SMA pueden entenderse como grupos de agentes que interaccionan entre sí para conseguir objetivos comunes. Una definición de un SMA se puede encontrar en [6], donde se plantea que un SMA es un sistema que reúne un ambiente, un conjunto de objetos, un conjunto de agentes que representan las entidades activas del sistema y un conjunto de operaciones que hacen posible el trabajo de los agentes sobre los objetos.

Tipos de prueba en agentes de software

Es difícil verificar que un agente o SMA satisfaga los requisitos del usuario. Una manera de ayudar a resolver este problema es desarrollar y usar patrones de diseño, así como algoritmos y herramientas para evaluar estos sistemas.

Realizar las pruebas en SMA es una tarea compleja debido a las características que ellos poseen, como son la autonomía, la proactividad y su habilidad de socializar con otros agentes. Existen tres tipos de pruebas para los SMA más desarrolladas, ella son: [7] Pruebas de agente, prueba de integración y prueba de sistema.

Otros autores consideran que deben realizarse un conjunto de actividades más relacionadas con las pruebas que no están declaradas explícitamente en ninguna metodología; estas actividades son: [8] Prueba de autonomía, prueba de proactividad, prueba de colaboración y prueba de cambio de estructura organizacional. Esto se debe a que las pruebas de agente, de integración y de sistema que asumen mucho de lo estipulado en la orientación a objetos. Pero los objetos no son autónomos, no son proactivos, no simulan a un ser humano como lo hacen los agentes.

Metodologías orientadas a agentes y pruebas

La mayoría de las metodologías orientadas a agentes no desarrollan completamente la etapa de prueba, algunas no la tienen en cuenta en su ciclo de vida. Estudios realizados por Moreno [8] muestran que de las metodologías PASSI, [9] INGENIAS, [10] TROPOS, [11] GAIA, [12] MaSE, [13] MESSAGE/UML, [14] Vowel Engineering, MAS-CommonKADS [3] y Prometheus, [15] solo PASSI propone la etapa de prueba pero no la desarrolla. Actualmente metodologías como Ingenias [16] y Tropo [17] han avanzado en el desarrollo del análisis de los errores.

La metodología PASSI propone que después del modelo de código se debe hacer una prueba de agente, y posterior al modelo del despliegue, hacer una prueba de integración, lo que en ambos casos solo dice que estas pruebas se deben hacer pero no cómo, ni qué artefactos obtener.

Durante los últimos años los creadores de la herramienta Case para la metodología PASSI (Passi ToolKit) han dirigido su investigación a consolidar dicha herramienta y crear un *framework* de prueba que suministre un uniforme acercamiento automatizado a la prueba de SMA. Con dicho *framework* se reduce el tiempo necesario para crear las nuevas pruebas y analizar los resultados.

Este *framework* es construido sobre la plataforma JADE y permite a diseñadores crear las pruebas en dos niveles de modelos: [18] en el primer nivel se identifica el agente como una entidad atómica, para verificar con exactitud las actividades llevadas a cabo por un solo agente, donde deben probarse varios casos diferentes; y en el segundo, se identifican las tareas específicas del agente.

La metodología Ingenias, según su autor, se puede llevar a acabo de igual manera que un software convencional, sin embargo, esto no se cumple así porque el comportamiento de un SMA es emergente y por tanto difícil de pronosticar.

Juan Botía, Jorge Gómez Sanz y Juan Pavón, [19] proponen la herramienta ACLAnalyser que integrada con Ingenias Development Kit (IDK), la herramienta para Ingenias permite verificar algunas propiedades de los SMA en el nivel de diseño donde la detección de errores es menos compleja que en el código del programa.

Por otra parte, Jorge Gómez Sanz, Rubén Fuentes y Juan Pavón [20] plantean: "La especificación detallada de un SMA puede definir la existencia de las pruebas", luego los diseñadores pueden utilizar las bibliotecas de la herramienta IDK para inspeccionar el estado interior de los agentes y rastrear cada comportamiento individual.

Los SMA giran sobre las conversaciones entre agentes, estas deben juzgarse correctamente antes de echar a andar el servicio. En [21] se propone integrar métodos a la metodología MaSE para verificar automáticamente que estas conversaciones sean válidas antes de emplearlas.

Para desarrollar estos métodos se utiliza la herramienta Case con que cuenta MaSE (AgentTool), la cual es totalmente dependiente; esto constituye una restricción, aunque tiene como ventaja según sus autores: su fácil manipulación argumentando que solo hay que experimentar con la misma. Primero se crean las conversaciones del agente usando los diagramas de la transición estatales gráficamente en la

herramienta. Esta representación gráfica se transforma entonces en un lenguaje de modelado formal llamado Promela que es analizado por la herramienta de prueba Giro para descubrir los errores como el bloqueo, ciclo, errores de la sintaxis, mensajes y estados sin usar. [21]

David Poutakidis, Lin Padgham y Michael Winikoff trabajan en propuestas para encontrar y remover errores de un agente software y han planteado mecanismos de especificación de protocolos disponibles, que a través del diseño van monitoreando la ejecución de un SMA. [22]

Las especificaciones de estos protocolos pueden ser usadas en tiempo de ejecución para reportar cualquier discrepancia dentro de las interacciones de los agentes en un SMA teniendo como patrón de comparación lo que fue especificado anteriormente; ellos se especifican utilizando AUML que se traduce a *Petri nets.* [23] Lo antes expuesto son las acciones de la metodología Prometheus para con las pruebas.

Desing tool (PDT) [24] es la herramienta que apoya a Prometheus, ella contiene un editor gráfico que soporta la especificación y diseño de tareas dentro de dicha metodología orientada a agentes. Esta herramienta propaga la información hasta donde le sea posible, y asegura la consistencia entre varias del diseño. [25]

La metodología Mas-CommonKADS por su parte, propone dentro de los modelos de proceso de software para proyectos pequeños, dos actividades encaminadas a las pruebas: Codificación y prueba, e integración y prueba global. Esta metodología establece que se puede comprobar parcialmente la corrección de la conducta global del sistema utilizando los escenarios típicos que tratan los posibles conflictos y los métodos de resolución de los mismos. Pero dado que la conducta global del sistema no puede ser determinada durante la fase de análisis o diseño, porque depende de los acuerdos y compromisos concretos realizados entre los agentes, normalmente se necesitará recurrir a la simulación.

La metodología Tropos por su parte propone la herramienta *eCAT* para dar soporte a las pruebas; esta herramienta la integran tres componentes principales: [7] Test Suite Editor: semiautomáticamente genera el esqueleto de colecciones de prueba y permite los verificadores humanos para especificar los datos de la prueba de los diagramas de análisis de meta producido por TAOM4E2, [11] una herramienta que apoya a Tropos. Autonomous Tester Agent: agente que es capaz de ejecutar las colecciones de la prueba contra un sistema multiagente. Monitoring Agent: supervisa comunicaciones y eventos entre agentes ayudando a la depuración.

Toolkit es la herramienta que soporta a la metodología Zeus para construir las aplicaciones de agentes colaborativos, integra la construcción rápida de aplicaciones de agentes. Dicha herramienta para encontrar y eliminar errores alterna la carga de inferencia desde el usuario hasta el visualizador. El visualizador es un agente que solo demanda la información local de otro agente e intenta construir la vista global.

PROPUESTA DE ACTIVIDADES DE PRUEBAS PARA INGENIAS

El V-Model

El V-Model es un modelo de procedimiento alemán para el desarrollo de software que prevé una secuencia fija de pasos de trabajo, que están divididos en una fase de desarrollo y una de comprobación. Como los pasos individuales de la fase de desarrollo son paralelos en el contenido con los de la fase de comprobación y se verifica su concordancia, gráficamente se produce una V.

La utilización del V-Model facilita la identificación de diferentes actividades y técnicas de pruebas y provee un marco de trabajo para revisar trabajos previos e identificar las necesidades de futuros encargos en algunas direcciones. [26]

La figura 1 muestra las fases del ciclo de vida del software en el lado derecho, y en el izquierdo, los tipos de pruebas que se corresponden con cada una de las fases.

La información para cada nivel de prueba es usualmente derivada de la actividad del desarrollo relacionada. Ciertamente, una importante recomendación a seguir es diseñar simultáneamente las pruebas con cada actividad del desarrollo, aunque el software no va a estar en un estado ejecutable hasta la fase de implementación.

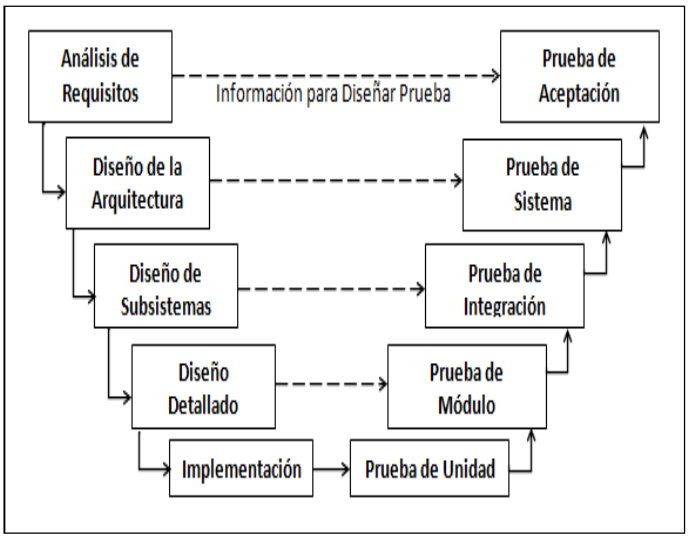


Fig. 1. Actividades del desarrollo y niveles de prueba en el V-Model. [26]

De forma general, los propósitos que persigue cada una de las pruebas de la rama derecha de la V son los siguientes:

- Prueba de aceptación: Determinar si el software final satisface los requisitos del sistema.
- Prueba de sistema: Determinar si el sistema congregado satisface las especificaciones.
- Prueba de integración: Valorar si las interfaces entre los módulos en un subsistema dado tienen suposiciones

coherentes y se comunican correctamente. Esta prueba debe suponer que los módulos trabajan correctamente.

• Prueba de módulo: Valorar un módulo individual aisladamente, incluyendo cómo las unidades de componentes interactúan con cada una de las otras y sus estructuras de datos asociadas.

• Prueba de unidad: A un bajo nivel, valorar las unidades producidas por la fase de implementación.

En este trabajo solo se propondrán las actividades referentes a las pruebas de aceptación y a los sistemas, ya que las demás actividades aún están en una etapa de desarrollo.

Actividades de pruebas para la metodología Ingenias

Ingenias es una metodología orientada a agentes, basada en la definición de una serie de metamodelos que describen los elementos que conforman al SMA desde varios puntos de vista, y permiten definir un lenguaje específico para el mismo. Esta metodología, como se vio en el epígrafe anterior, cuenta con una herramienta (ACLAnalyser) que realiza pruebas muy simples que son insuficientes.

Pero como el proceso de instanciación de los metamodelos no resulta trivial, puesto que existen muchas entidades y relaciones a identificar, además de las dependencias existentes entre ellos, Ingenias define un grupo de tareas cuya ejecución termina en un conjunto de estos modelos. Estas actividades se enmarcan en flujos de trabajo, los cuales persiguen objetivos en cada una de las fases en que son divididos. En la tabla 1 se muestran cada uno de estos flujos de trabajo y las actividades que en cada una de las fases se han de llevar a cabo.

Traspolando estas fases de elaboración y haciéndolas coincidir en el modelo para el desarrollo de software V-Model, se obtiene un V-Model para la metodología Ingenias, el cual se muestra en la figura 2.

Una vez confeccionado el modelo de desarrollo para la metodología Ingenias se está en condiciones de definir los objetivos que se han de cumplir para cada una de las etapas, así como cada una de las actividades de prueba que se han de llevar a cabo dentro de las mismas.

A continuación se definen los objetivos de algunas de las etapas de pruebas, así como las actividades de pruebas, las que son necesarias realizar en cada momento.

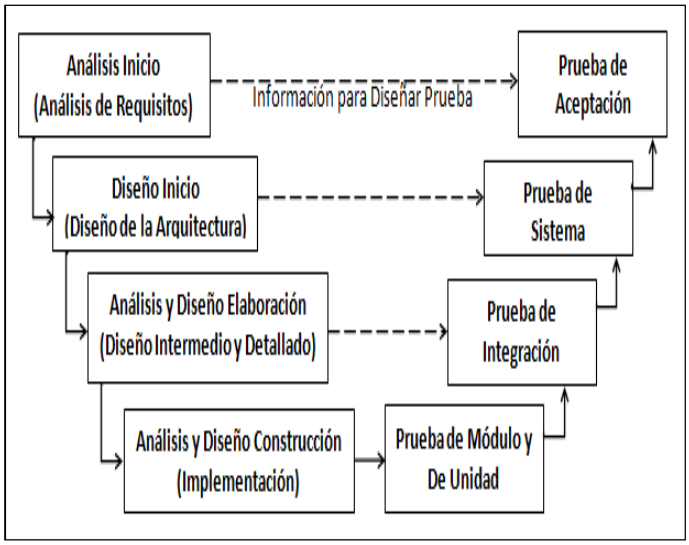


Fig. 2. V-Model para la metodología Ingenias.

Tabla 1
Descripción literal para el caso de uso *Gestionar Información*

Caso de Uso	Gestionar Información *
Precondiciones	El usuario tiene que haber hecho una solicitud de información, su perfil ya está cargado y ya tiene asignado un AT
Resumen	El CU se puede iniciar de dos formas: Una cuando el A P le solicita al AT la información que el usuario le pidió a través del ACE, luego el AT se va comunicar con el ACP que se encarga de pedirle la información a la PPT Cuando la PPT le entrega la información al ACP este se la da al AT que se la pidió y este a su vez se la entrega al AP que a través del ACE se la hace llegar al usuario Otra manera de iniciar el CU es cuando el AP se comunica con el AT porque el usuario desea subir una información. El AT se comunica con el ACP para entregar la información a la PPT que se encarga de organizarla y clasificarla según la información que se desea subir. Una vez realizada la acción la PPT le responde al ACP que la gestión se realizó correctamente. El ACP le responde al AT, que este se comunica con el AP y este último a través del ACE con el usuario para comunicarle que la información ya está publicada
CU Asociados	Solicitar información y subir recurso de información

* Ver epígrafe "Descripción de los artefactos propuestos..." para conocer el significado de las siglas.

Revista Cubana de Ingeniería. Vol. III, No. 3, septiembre - diciembre, 2012, pp. 47 - 56, ISSN 2223 -1781

Prueba de aceptación

La fase de análisis de requisitos es la encargada de recopilar las necesidades del usuario. Por lo que la prueba de aceptación se diseña para determinar si el software, una vez completado, satisface estas necesidades, en otras palabras, esta prueba se realiza para conocer si el software hace lo que los usuarios quieren. Esta prueba ha de involucrar a usuarios y otros individuos que tengan un fuerte conocimiento del negocio.

En la metodología Ingenias no se hace referencia de manera explícita, a la captura de requisitos. En contraste con esto, la actividad número uno en la etapa de análisis del proceso que Ingenias propone, es la identificación de los casos de uso del sistema, como los requisitos más importantes del sistema, y a partir de ellos se realizarían técnicas de generación de pruebas de requisitos enfocadas a probar el cumplimiento de los mismos.

El objetivo fundamental de las actividades de pruebas en esta fase de aceptación es preparar las pruebas de aceptación y las prueba de sistema; las pruebas de aceptación en menor medida. Las siguientes actividades llevan a cabo estos objetivos:

- Criterio de prueba.
- Soportes de software.
- Plan de prueba.

Las actividades anteriores, no son soportadas por las metodologías orientadas a agentes, pero esto no implica un gran problema. Las mismas pueden ser adoptadas fácilmente, porque no necesitan estar soportadas por una herramienta. Los artefactos obtenidos en estas actividades ayudan a los probadores del equipo de proyecto en el proceso de prueba.

Prueba de sistema

La fase de diseño de arquitectura en cualquier proceso de desarrollo de software es la guía para el desarrollo del mismo, en esta fase se diseñan los componentes de la aplicación lo cual permite visualizar la interacción de las entidades del negocio. De forma general, en esta fase se describe cómo se construirá la aplicación de software.

En la metodología Ingenias este eje de desarrollo lo constituye el Modelo de Organización, el cual se obtiene desde la primera fase de ejecución.

Por lo tanto, el objetivo principal de la actividad de prueba en esta fase, se centraría en el diseño de la arquitectura, y sería validar el mapeo entre las especificaciones de los requisitos y el diseño. La actividad más importante en esta etapa es validar el diseño. En la actividad de validar el diseño es importante chequear la correlación entre las metas del sistema, y la capacidad y roles del agente. Esto es diferente con respecto a la orientación a objetos. [27]

El propósito principal de estas actividades de prueba es preparar la prueba de sistema (con menos énfasis) para la prueba de aceptación y para las pruebas de unidad e integración. Prueba de diseño del sistema, criterio de cobertura de desarrollo y diseño de plan de prueba de aceptación, son actividades que cumplen este propósito.

Las metodologías orientadas a agentes no presentan este tipo de prueba. Sin embargo, el diseño de plan de prueba de aceptación y el criterio de cobertura de desarrollo son actividades que pueden ser llevadas a cabo manualmente y por tanto se pueden adoptar fácilmente. Los artefactos obtenidos en estas actividades ayudan a los probadores del equipo de proyecto en el proceso de prueba.

La estructura organizacional es importante para la prueba de diseño del sistema; esta es diferente con respecto a la orientación a objeto. La prueba de diseño del sistema y validar el diseño necesitan una herramienta de soporte y las mismas requieren un estudio detallado para ser adoptadas en una metodología orientada a agentes.

DESCRIPCIÓN DE LOS ARTEFACTOS PROPUESTOS PARA CADA NIVEL DE PRUEBA

Mediante el caso de estudio descrito se desarrollará el flujo de pruebas propuesto, que se refiere a un observatorio tecnológico (OT).

La arquitectura del SMA que respaldará un OT y contará con una serie de agentes que desempeñarán diferentes funcionalidades: Agente comunicador externo (ACE), agente personal (AP), agente temático (AT), agente controlador *blackboard* (ACB) y agente comunicador plataforma (ACP).

Se dispone de un Portal Corporativo que se comunicará con el OT, dicho sistema es externo al OT y será el encargado de autenticar a los usuarios para que accedan a los servicios que él brinda. Esta aplicación se comunicará con el OT a través del ACE que mantendrá al sistema en contacto con el Portal u otra aplicación externa.

Las solicitudes de información que pide un usuario serán recogidas por un AP que se encargará de gestionar el perfil del usuario, para que la información que se brinde al mismo responda a sus intereses. Una vez gestionado el perfil, el AP se comunicará con el AT que controle el tema sobre el cual el usuario está solicitando información, para responder a las solicitudes realizadas al OT.

El AT se comunicará con el ACP que se conecta con los servicios de una plataforma de procesamiento de texto (PP T), sistema externo al OT, que se encargará de llevar a cabo todo el proceso de clasificación y almacenamiento de la información. La PPT le brinda a los AT la información solicitada de manera organizada.

Es importante tener en cuenta que varios A P pueden estar suscritos a varios AT con el objetivo de compartir información de interés común entre varios AP. Existen tantas instancias de este agente como macrotemas se vayan agregando a la plataforma. Cada uno tendrá así mismo, varios AP que atender.

El OT tendrá un agente controlador que será el encargado de supervisar toda la comunicación dentro del SMA, accediendo para ello a una base de datos que seguirá el patrón arquitectural *Blackboard,* [28] respondiendo a las preguntas del tipo *¿qué agente sabe*? o *¿qué agente tiene información sobre*?. Este agente debe guardar las trazas de todos los mensajes enviados entre el resto de los agentes del sistema, de esta manera se tiene una base de datos de la comunicación existente en la plataforma, de manera que se puede obtener información vital para el funcionamiento del OT.

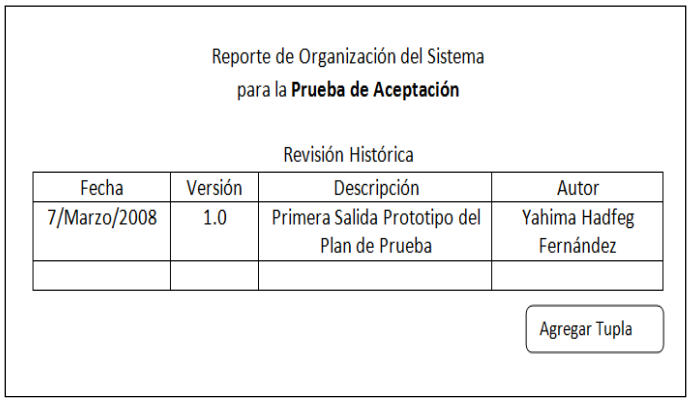
Prueba de aceptación

En la prueba de aceptación se genera una entidad llamada plan de prueba inicial. Para representar esta entidad, se propone un artefacto haciendo uso de la técnica Web, donde se almacenen en un fichero los aspectos fundamentales (identificador único del documento, históricos, introducción, resumen de elementos y características a probar, casos de usos del sistema, elementos a probar y características a probar). Esta página Web tiene como característica, que el usuario vaya interactuando con la misma los campos pertinentes. Es de señalar que existirán algunos aspectos que el usuario no debe llenar, ya que la herramienta debe de generar automáticamente.

La figura 3 muestra dónde se lleva el registro histórico o trazas, que se irían llenando para tener constancia del personal que va revisando este documento. La página cuenta además, con unos enlaces directos, para facilitar la rápida navegación. Estos se muestran a continuación.

Tabla de contenidos

1. Introducción.
1.1. Propósito.
1.2. Enfoque.
1.3. Definiciones, acrónimos y abreviaturas.
1.4. Referencias
2. Resumen de los resultados de la prueba.
3. Lista de errores encontrados.
4. Acciones sugeridas. (Un ejemplo se muestra en la figura 3).


Reporte de Organización del Sistema
para la **Prueba de Aceptación**

Revisión Histórica

Fecha	Versión	Descripción	Autor
7/Marzo/2008	1.0	Primera Salida Prototipo del Plan de Prueba	Yahima Hadfeg Fernández

Agregar Tupla

Fig. 3. Muestra de históricos del plan de prueba inicial.

Entre los elementos más importantes que se muestran en la página se encuentran los referentes a los objetivos, los casos de uso del sistema, los requisitos para las pruebas y la estrategia de pruebas. Dentro de los primeros mencionados, se encuentran aspectos importantes como la introducción y resumen de elementos, así como las características a probar y el enfoque general de la prueba.

Los casos de uso del sistema que se obtienen son: Gestionar perfil, subir recurso de información, solicitar información, gestionar información, controlar comunicación y gestionar trazas. A continuación se muestran algunos de los requisitos para las pruebas asociadas al caso de estudio.

2. Requisitos para las pruebas

La lista que se muestra a continuación identifica los elementos (casos de uso, requisitos funcionales y requisitos no funcionales) que fueron identificados como blanco de prueba. Esta lista representa qué sería probado.

2.1. Elementos de software a probar

Verificar la correcta realización del caso de uso "gestionar información".

2.2. Características a probar

- Validar la organización
- Validar prototipo
- Validar el diseño

2.3. Características que no se probarán

Demás casos de uso

2.4. Documentos a entregar

- Plan de prueba inicial
- Plan de prueba de aceptación
- Reporte de organización
- Reporte de diseño
- Diagrama de actividad resultante de la actividad validar prototipo.

2.5. Soporte de software

Ingenias Development Kit (IDK)

2.6. Criterio de prueba general

Prueba de sistema

Siguiendo las actividades propuestas para la prueba de sistema se sugiere partir de una lista de chequeos y la descripción literal de los casos de uso, en la actividad validar el prototipo, obtener un diagrama de colaboración describiendo la secuencia de pasos que el caso de uso debe atravesar una vez activado; la descripción literal para el caso de uso "gestionar información" que se encuentra en la tabla 1.

Continuando con la secuencia de actividades descritas en este artefacto obtenido a partir del IDK, se logra el diagrama de colaboración que se muestra en la figura 4.

Como resultado final de la fusión de las actividades definidas anteriormente, que involucra a las actividades de validar prototipo y generar plan de pruebas aceptación, se propone generar el plan de prueba de aceptación e incorporar los

diagramas presentados anteriormente en dicha entidad. Para representar la entidad, se propone un artefacto dinámico con la técnica Web, donde se almacenen en un fichero los aspectos importantes.

La actividad validar organización está destinada a encontrar errores en la organización del sistema. Esta actividad genera una entidad llamada reporte organización encaminado a detallar los errores encontrados en la organización del sistema. Para representar esta herramienta se propone utilizar una página Web que se guarda en un fichero para dejar constancia de la revisión de esos errores.

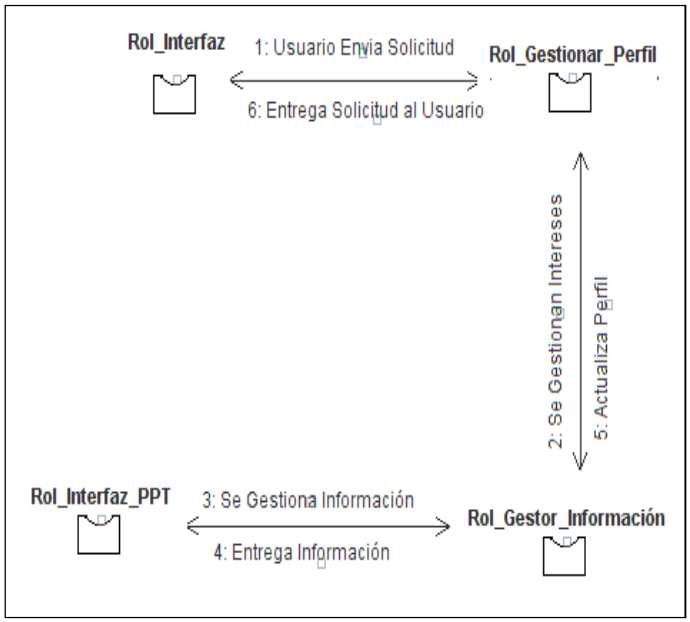


Fig. 4. Diagrama de colaboración caso de uso "Gestionar Información" con Ingenias.

Para el caso de uso que se está analizando, la página mostraría la lista de los errores encontrados, la lista de acciones a emprender para resolver los errores y los diagramas dónde se originan los errores; esta información será mostrada por la herramienta, de la siguiente forma:

3. Lista de errores encontrados

Organización "Grupo Personal" falta "Rol_Subir_Información".

4. Acciones sugeridas

Verificar Diagramas de Agentes "A.P", adicionar "Rol_Subir_Información" a la organización "Grupo Personal"

En las figuras 5 y 6 se muestran los diagramas dónde se encontraron los errores mencionados:

Como se aprecia, el agente AP tiene definido Rol_Subir_Informacion. Este mismo agente, que pertenece a la organización "Grupo Personal", el Rol_Subir_Informacion no se encuentra en la misma. Esto constituye un error de la organización, puesto que los roles de los agentes, deben estar definidos en la organización a la que pertenecen y sin embargo, el IDK no reconoce este error.

La actividad validar diseño está destinada a encontrar errores en el diseño del sistema, que usualmente estos errores se encuentran asociados al entorno. Esta actividad genera una entidad llamada reporte diseño que posee una estructura similar al reporte organización. Para representar esta herramienta también se utilizó una página Web que guarda en un fichero los datos generados, dejando constancia de la revisión de esos errores.

La técnica utilizada para desarrollar la actividad antes mencionada fue el traceo de los diagramas fundamentalmente los de análisis, ya que esto garantiza que no falte ninguna de las especificaciones en el diseño. Esto se debe a que todos los artefactos del análisis deben tener un respaldo en el diseño.

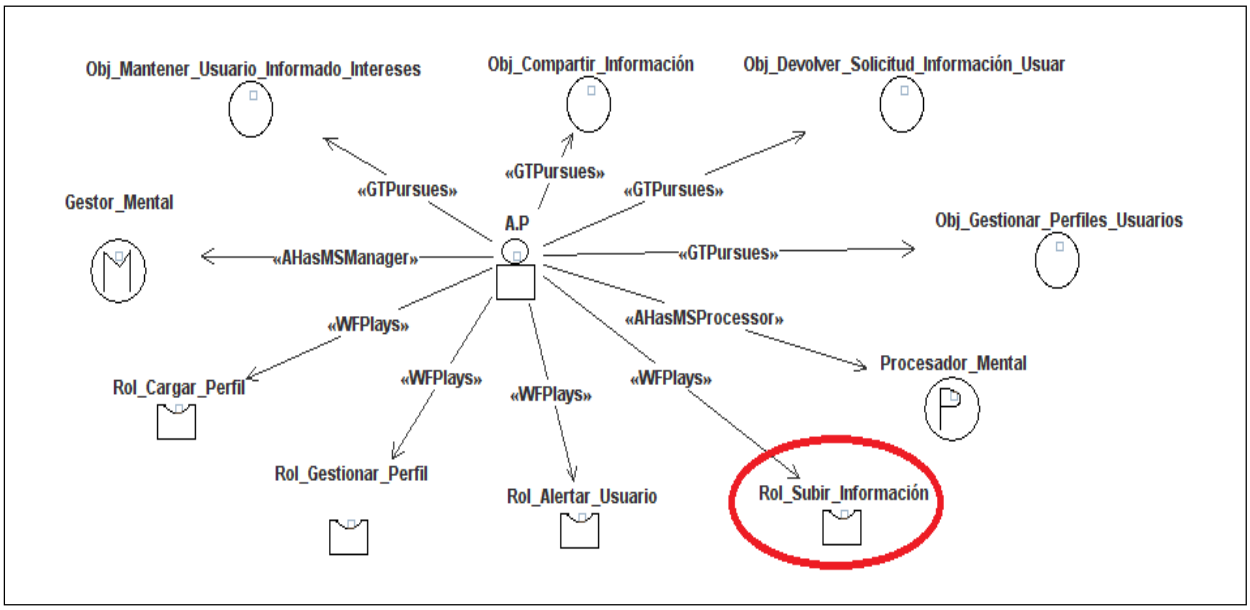


Fig. 5. Diagrama Agente "Agente Personal".

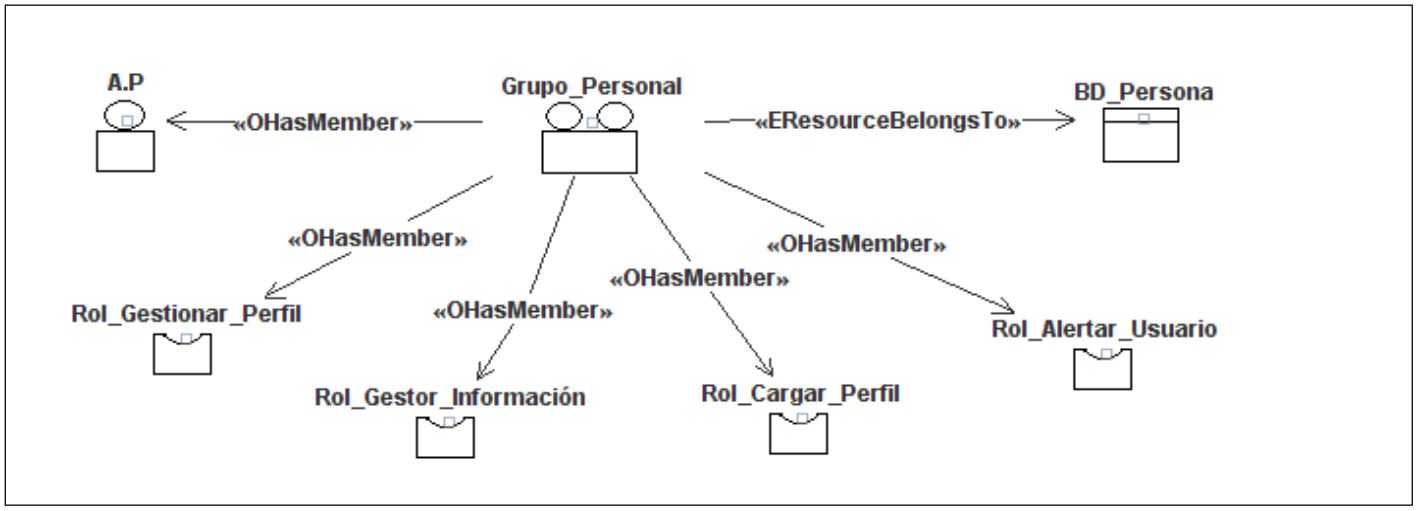


Fig. 6. Diagrama de organización "Grupo Personal".

CONCLUSIONES

En este trabajo se recopilaron datos sobre el trato dado a la fase de prueba por algunas de las metodologías orientadas a agentes. Se puede observar que la mayoría de las metodologías en su proceso no exponen explícitamente qué actividades seguir para este proceso e incluso en algunos casos los autores plantean que el proceso de pruebas debe realizarse como mismo se realiza en las metodologías orientadas a objeto. No obstante, se observan adelantos por parte de algunas metodologías que presentan herramientas especializadas para realizar estas tareas, aunque hay que reconocer que la mayoría solo se dedica a hacer exploraciones muy simples del comportamiento de un SMA.

El V-Model brinda la posibilidad de identificar en cada etapa de desarrollo de un software el tipo de prueba que se le recomienda realizar al futuro sistema. Sobre esa base se pudieron proponer varias actividades de este tipo, que responden a las pruebas de aceptación y a las pruebas de sistema para la metodología Ingenias, siendo estas un paso de avance para llegar a contar con una fase de prueba para dicha metodología robusta.

Conviene señalar que algunos aspectos no fueron ampliamente tratados, como por ejemplo, el tratamiento a las pruebas de proactividad, que es uno de los aspectos donde existe más debilidad actualmente.

Se definen un conjunto de tareas y actividades de pruebas en las fases iníciales de la metodología Ingenias que permitirá mejorar el proceso de desarrollo de software guiando al usuario en la producción de un software libre de errores y con un adecuado funcionamiento. Estas tareas o actividades de prueba propuestas en las fases iníciales solventan la carencia de elementos de prueba en la metodología Ingenias

REFERENCIAS

1. BARNES, NOBLE. *A Guide to the Project Management. Body of Knowledge*. 2009. ISBN: 978-1933890517.

2. BOOCH, Grady; MAKSIMCHUK, Robert. *Object-Oriented Analysis and Design with Applications*. San Francisco: Addison-Wesley. 2007, 691 pp. ISBN 9780201895513.

3. HENDERSON-SELLERS, Brian; GIORGINI, Paolo. *Agent-Oriented Methodologies*. Hershey - London - Melbourne - Singapore: Editorial Idea Group Publishing, 2005, 420 pp. ISBN 1-59140-587-4.

4. WOOLDRIDGE, Michael; JENNING, Nicholas, "Intelligent Agents: Theory and Practice". *The knowledge Engineering Review*. 1995, vol. 2, pp. 115-152.

5. FIPA. Foundation for Intelligent Physical Agents. 2011 [Ref. 2011 febrero]; Disponible en Web: http://www.fipa.org/ [consultado en enero 2012].

6. WOOLDRIDGE, Michael. *An Introduction to MultiAgent Systems England*: John Wiley and Sons Ltd. 2002, 348 pp. ISBN 0-471-49691-x.

7. NGUYEN, Duy Cu; PERINI, Anna. "A Goal-Oriented Software Testing Methodology". *LNCS.* 2008, vol. 2, pp. 58-72.

8. MORENO, Mailyn. "Metodologías orientadas a agentes: un estudio comparativo". Director: Alejandro Rosete Suárez y Ailyn Febles Estrada. Tesis de Maestría, Cujae, La Habana, 2006.

9. CABRERA-PANIAGUA, Daniel; CUBILLOS, Claudio. "PASSI Methodology in the Design of Software Framework: A Study Case of the Passenger Transportation Enterprise". *LNCS*, 2009, pp. 213-227.

10. GÓMEZ-SANZ, Jorge. "Modelado de sistemas multiagentes". Director: Francisco Garijo Mazario y Juan Pavón Mestras. Tesis Doctoral, Universidad Complutense de Madrid, Madrid, 2002.

11. MORANDINI, Mirko; NGUYEN, Duy Cu. "Tool-Supported Development with Tropos: The Conference Management System Case Study". *LNCS*. 2008, pp. 182-196.

12. CERNUZZI, Luca; MOLESINI, Ambra. "Adaptable Multi-Agent Systems: The Case of the Gaia Methodology". *International Journal of Software Engineering and Knowledge Engineering*. 2011, vol. 4, pp. 491-521.

13. DELOACH, Scott; LUCK, Michael. "Developing a Multiagent Conference Management System Using the O-MaSE Process Framework". *LNCS*. 2008, vol. 4951, pp. 168-181.

14. CAIRE, Giovanni; COULIER, Wim. "Agent Oriented Analysis Using MESSAGE/UML". *LNCS*. 2002. pp. 119-135.

15. PADGHAM, Lin; THANGARAJAH, John. "The Prometheus Design Tool - A Conference Management System Case Study". *LNCS*. 2008, vol. 4351, pp. 197-211.

16. SERRANO, Emilio; SANZ, Jorge. "Intelligent Data Analysis Applied to Debug Complex Software Systems". *Neurocomputing*. 2009, vol. 72, pp. 13-15.

17. NGUYEN, Duy Cu; PERINI, Anna. "eCAT: a Tool for Automating Test Cases Generation and Execution in Testing Multi-Agent Systems". in *7th Int. Conf. on Autonomous Agents and Multiagent Systems* (AAMAS 2008). 2008, pp. 1673-1674

18. CAIRE, G.; Cossentino, M. "Multi-Agent Systems Implementation and Testing". in *Fourth International Symposium: From Agent Theory to Agent Implementation*. 2004, pp. 14-16.

19. BOTÍA, Juan; SANZ, Jorge. "Intelligent Data Analysis for the Verification of Multi-Agent Systems Interactions". *LNCS*. 2006, vol. 4224, pp. 1207-1214.

20. GÓMEZ, Jorge; FUENTES, Rubén. "INGENIAS Development Kit: a Visual Multi-Agent System Development Environment". in 7th Int. *Conf. on Autonomous Agents and Multiagent Systems* (AAMAS 2008). 2008, pp. 1675-1676.

21. LACEY, Timothy; DELOACH, Scott A. "Automatic Verification of Multiagent Conversations". In *The Seventeenth National Conference on Artificial Intelligence*. 2000, pp. 8.

22. POUTAKIDIS, David; PADGHAM, Lin. "An Exploration of Bugs and Debugging in Multi-agent Systems". In *Conf. on Autonomous Agents and Multiagent Systems* (AAMAS 2003). 2003, pp. 1100-1101.

23. POUTAKIDIS, David; PADGHAM, Lin. "Debugging Multi-Agent Systems Using Design Artifacts: The Case of Interaction Protocols". In *Conf. on Autonomous Agents and Multiagent Systems* (AAMAS 2002). 2002, pp. 960-967.

24. ZHANG, Zhiyong; THANGARAJAH, John. "Automated Unit Testing Intelligent Agents in PDT". In *7th Int. Conf. on Autonomous Agents and Multiagent Systems*. 2008, pp. 1673-1674.

25. THANGARAJAH, John; PADGHAM, Lin. "Prometheus Design Tool". In *Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2005)*. 2005, pp 127-128.

26. BECKER-KORNSTAEDT, Ulrike; VERLAGE, Martin. "The V-Modell Guide: Experience with a Web-Based Approach for Process Support". In *STEP '99 Proceedings of the Software Technology and Engineering Practice*. 1999, pp. 161-169.

27. MORENO, Mailyn; PAVÓN, Juan. "Testing in Agent Oriented Methodologies". *LNCS*. 2009, pp. 138-145.

28. DONG, Jing; CHEN, Shanguo. "Event-Based Blackboard Architecture for Multi-Agent Systems". In *Proceedings of the International Conference on Information Technology: Coding and Computing* (ITCC'05). Washington, 2005, pp. 379-384.

AUTORAS

Yahima Hadfeg Fernández
Ingeniera Informática, Máster en Ciencias, Instructora, Departamento de Inteligencia Artificial e Infraestructura de Sistemas Informáticos, Facultad de Ingeniería Informática, Instituto Superior Politécnico José Antonio Echeverría, Cujae, La Habana, Cuba

Mailyn Moreno Espino
Ingeniera Informática, Máster en Ciencias, Profesora Auxiliar, Facultad de Ingeniería Informática, Instituto Superior Politécnico José Antonio Echeverría, Cujae, La Habana, Cuba

Martha Dunia Delgado Dapena
Ingeniera Informática, Doctora en Ciencias Técnicas, Profesora Titular, Facultad de Ingeniería Informática, Instituto Superior Politécnico José Antonio Echeverría,Cujae, La Habana, Cuba

Approach Testing Activities for Ingenias

Abstract

This paper gathers the state of the phase of tests in the Agents and Multi-Agents systems for different methodologies that exist now a day. In its development, different kinds of tests that must be carried out on the Multi-Agents systems, will be explored; as well as what different Agent-Oriented methodologies propose to cover this important phase. Moreover, main characteristics of the tools that support the design of these methodologies are exposed; as well as modules or mechanisms that have been incorporated into them in order to show the direct impact such problem has on the quality of software produced. Ingenias is one of the Agent-Oriented methodologies that do not include a phase for functional test. In this paper, Acceptance and Systems Tests are proposed for this methodology, based on V-Model, allowing the construction of software with a higher quality.

Key words: agent, agents' test, ingenias, V-Model, acceptance tests, system tests