Integración de agentes inteligentes heterogéneos en el entorno T-World

Sergio Burdisso, Guillermo Aguirre, and Marcelo Errecalde

Universidad Nacional de San Luis, Departamento de Informática
Ejército de los Andes 950, San Luis.
sergio.burdisso@gmail.com,gaguirre@unsl.edu.ar,merreca@unsl.edu.ar

Resumen A fin de facilitar la tarea de crear y experimentar con agentes computacionales inteligentes se ha creado una plataforma que funciona como campo de experimentación portable. Cuenta con una simulación gráfica atractiva y numerosas cualidades que la hacen ideal para docencia. Uno de los desafíos más importantes es manejar la heterogeneidad de los agentes, lo que se consigue mediante un programa intermediario.

1. Introducción

Una ejercitación típica en los cursos de IA consiste en plantear distintos tipos de situaciones para las cuales los alumnos deben encontrar una solución, codificar esa solución en algún lenguaje de programación y finalmente evaluar cómo se desempeña el agente creado. Existen pocos escenarios controlados [3] donde los estudiantes puedan desarrollar este tipo de ejercitación. Las limitaciones provienen por las restricciones del lenguaje de programación empleado y por la escasez de variantes en los parámetros de configuración del entorno de prueba. Uno de los pocos recursos de este tipo que ha sido bastante reconocido es Tileworld [5], pero este mundo simulado presenta numerosas dificultades que han impedido su uso de manera masiva. Muchos de estos impedimentos se describen en un trabajo previo [2].

Actualmente se encuentra en funcionamiento una plataforma de experimentación denominada T-World (tworld-ai.com) que permite configurar escenarios similares a los de Tileworld, crear agentes para resolver los problemas planteados y verificar el funcionamiento de los mismos mediante una simulación gráfica. Para facilitar la accesibilidad y simplicidad de utilización, se optó por utilizar el navegador web como plataforma de implementación. Esto aumenta la portabilidad del sistema debido a que todos los dispositivos actuales tales como smartphones, tablets y computadoras personales disponen de un navegador web. Además permite que el sistema pueda ser accedido vía Internet (o localmente) y ejecutado directamente en el dispositivo del usuario sin necesidad de instalar o configurar software adicional.

El campo de pruebas construido es una extensión de Tileworld, consiste en una grilla bidimensional en la cual existen agentes, huecos, baldosas y obstáculos. El robot puede moverse en las cuatro direcciones del plano y si está junto a una

baldosa la puede empujar. Los obstáculos no pueden ser atravesados por los robots y los huecos deben ser tapados con baldosas, el objetivo es llenar tantos huecos como sea posible. Este entorno es dinámico ya que los huecos aparecen y desaparecen aleatoriamente lo que representa todo un desafío a la hora de implementar algoritmos que resuelvan las pruebas.

La interfaz gráfica de usuario cuenta con una gran colección de elementos visuales tales como menús, íconos, botones e imágenes junto al entorno en tres dimensiones para las simulaciones. Desde el comienzo del diseño del T-World quedó claro que la comunicación con los agentes tendría que ser bidireccional, ya que tanto el ambiente como los agentes pueden recibir respectivamente acciones y percepciones, sin seguir ningún patrón preestablecido, e.d. ambos pueden hacerlo en cualquier momento y en cualquier orden, incluso en simultáneo. Sin embargo, este requerimiento se contrapone a la idea original con la cual fue concebida la Web, donde el usuario especifica una URL para recuperar un recurso y como respuesta el servidor entrega una instancia del mismo —por ejemplo, un documento HTML, un archivo de imagen o de cualquir otro tipo. Afortunadamente, con la reciente aparición de WebSocket de HTML5, ahora sí es posible tener un modelo de comunicación full-duplex para el navegador web. El protocolo *WebSocket* proporciona canales de comunicación *full-duplex* sobre una única conexión TCP.

En este artículo se describe un componente que es clave para permitir que en T-World se ejecuten agentes codificados en diferentes lenguajes de programación, este componente es denominado *proxy*. Mediante el proxy se vinculan, de una manera transparente, los robots que corren en la simulación con los agentes creados por los alumnos. Las percepciones del entorno y las acciones sobre el mismo son realizadas por los agentes respetando las características particulares del lenguaje de programación escogido. Es el proxy quien se encarga de hacer la traducción para cumplir con el formato de protocolo requerido para interactuar con T-World.

2. Entorno T-World

El diseño e implementación de *T-World* se hizo para tener una versión de Tileworld que los estudiantes pudieran usar inmediatamente. Desde el plan inicial se pretendió que fuese *portable* y *flexible*, y que permitiera incorporar razonamiento a los robots mediante *programas* escritos en cualquier lenguaje. También se consideró la aplicación de diversos parámetros al mismo tiempo y que la simulación fuese *visual*, para lograr experiencias educativas atractivas.
En la Figura 1 se observa que la experiencia es similar a un experimento real en el contexto de un juego, ya que por ejemplo se puede ver el efecto del robot al empujar las baldosas. También se puede observar la reacción del robot cuando choca con un obstáculo.

T-World además de incorporar un editor de código para programar los robots, brinda una API con facilidades para desarrollarlos.
El diseño de la plataforma se basó en las ideas y conceptos presentados en el libro

Figura 1: Captura de pantalla de T-World

Artificial Intelligence: A modern approach [6], en esta sección se abordan algunos de esos fundamentos con la intención de explicar cómo se relacionan entre sí y cómo fueron tenidos en cuenta durante la implementación de T-World.

2.1. Descripción del agente

Un agente es todo aquello que se pueda ver como una entidad que percibe su ambiente a través de *sensores* y actúa sobre ese ambiente a través de *actuadores* [6]. Ver figura 2. El término percepción se refiere a la entrada recibida por el agente en un determinado momento. Una secuencia de percepciones de un agente es la historia completa de todo lo que el agente ha percibido. En general la elección de la acción que hace el agente en un instante dado, depende sólo de la secuencia de percepciones observadas y no de lo que no ha percibido.
En el contexto de T-World los agentes se corresponden con los *robots* de la simulación 3D, ver figura 1.

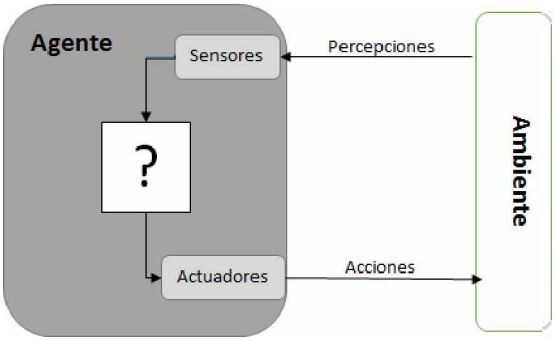


Figura 2: Interacción con el ambiente a través de sensores y actuadores

2.2. La percepción

Los agentes perciben el ambiente en el que se encuentran por medio de *sensores*, un agente humano percibe mediante sus ojos, sus oídos, etc., un robot lo hace por medio de cámaras y sensores infrarrojo. En T-World un programa agente recibe el estado del ambiente cuando es creado y posteriormente por requerimientos explícitos invocando a la función *$perceive()*. Aquí, una percepción brinda la información que necesita el agente para tomar sus decisiones, por ejemplo, su próxima acción. En T-World la percepción que es enviada hacia el agente está compuesta por dos partes: el *encabezado* y los *datos* propiamente dichos. En el encabezado, el ambiente especifica si se trata de un objeto con información sobre el estado del ambiente o si se trata de un evento especial como un mensaje o la finalización de la simulación. La parte de datos se distribuye en tres compartimentos según el origen que tenga: Información *externa* al agente (e.g. disposición de elementos físicos y de otros agentes en el mundo, tiempo transcurrido, etc.). Información *relativa al agente* receptor de la percepción (niveles de energía, puntuación, posición actual, etc). Y el tercer tipo de información corresponde al *conocimiento incorporado*, son los valores constantes establecidos por el usuario para ese ambiente en particular. Por ejemplo dimensiones del escenario, los valores y las reglas que definen cómo evoluciona el ambiente.

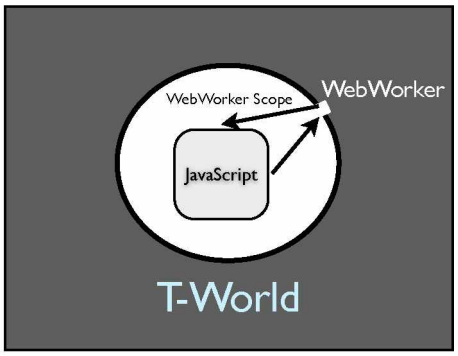
2.3. Las acciones

La descripción del comportamiento de un agente se completa con la especificación de sus acciones. Los agentes actúan sobre el ambiente por medio de sus *efectores*, lo que para un agente humano serían sus manos, piernas y cuerdas vocales, para un agente de software pueden ser, hacerse visible en la pantalla, escribir en un archivo y enviar un paquete por la red. Los agentes que están inmersos en T-World actúan por medio de las siguientes acciones: *desplazamiento* en la grilla, *restauración* de la batería y *envío* de mensajes a otros agentes. El desplazamiento consiste en actualizar la ubicación del agente a una nueva fila o columna adyacente a la actual, siempre que sea posible hacerlo. La restauración de la batería se puede hacer cuando el agente se ha quedado sin carga; antes de realizar esta acción el agente debe considerar su costo, ya que a cambio de la energía deberá ceder la mitad de su puntaje actual. Un agente puede enviar un mensaje a un agente específico o difundirlo a todos los integrantes del equipo. El contenido de un mensaje es una secuencia de caracteres que idealmente deberá estar convenientemente estructurada, por ejemplo, siguiendo los estándares de comunicación entre agentes como la establecida por FIPA [1]. Existe una acción especial que es hacer *nada*, empleada por ejemplo para indicarle a T-World que el agente necesita más tiempo para decidir la próxima acción o cuando no es posible realizar ninguna otra acción.

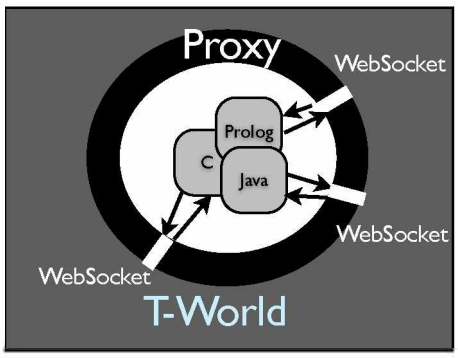
2.4. Programas Agente

El comportamiento de un agente se describe por una *función agente*, que a cada secuencia de percepciones le hace corresponder una acción. Se podría

pensar en crear una tabla equivalente a la función agente para describir cualquier agente. Para eso sería necesario tabular todas las secuencias posibles de percepciones y para cada una de ellas registrar la acción escogida por el agente. La tabla es claramente una caracterización externa del agente. Internamente, la función agente será implementada por un *programa agente*. El programa agente se corresponde con el signo de interrogación en la figura 2.
En T-World, dependiendo del lenguaje usado para codificar el programa agente se consideran dos situaciones diferentes: cuando se usa JavaScript y cuando se usa algún otro lenguaje que no sea JavaScript, como ser Java, C o Prolog.
En T-World la creación de un programa agente consiste de los siguientes pasos:
1) Asignar un nombre al programa agente, esto es necesario entre otras cosas, para que el usuario identifique a su programa agente dentro del sistema.
2) Determinar cómo se controlará el robot, las posibilidades son que sea el usuario humano mediante el teclado o algún algoritmo escrito en un lenguaje de programación. El T-World procede de manera distinta cuando el lenguaje usado para el agente es JavaScript o cuando es algún otro lenguaje, este último caso se describe en la próxima subsección.
3) Ingresar una descripción del programa agente, útil para ayudar a reconocer las características del programa agente como el algoritmo escogido para razonar.



3 a) Lenguaje JavaScript



3 b) Distintos lenguajes

Si se ha elegido escribir el programa agente en JavaScript, en el segundo paso se abre automáticamente un editor con un esqueleto básico para comenzar a programar. Dentro del editor se presentan cuatro secciones de código: la primera, "Agent Program", es para el algoritmo principal del programa agente. La segunda, "Global Scope", se usa para definir funciones y variables de alcance global. La tercera sección, "Start Event", es para manejar el evento de "inicio de simulación" que se dispara al comenzar la simulación. Esto es útil para inicializar variables o estructuras de datos que dependan de la primer percepción y que luego usará el agente. Generalmente se usa para obtener la configuración inicial

del ambiente: condiciones de finalización, tamaño de la grilla, etc. La última sección, "Message Received Event", se debe implementar solamente cuando el agente requiere recibir mensajes, o sea, cuando integra un sistema multi-agente.

Después de crear el programa agente ya puede comenzar a interactuar con T-World. Se debe asociar una configuración del ambiente de tarea con el programa agente creado, esto se puede hacer directamente desde el ambiente del editor.

El programa agente se ejecuta como un Web worker (ver figura 3a), es decir como un script en el background independiente del script principal que atiende la simulación. La invocación al constructor Worker() crea un worker y retorna un objeto Worker que representa al programa agente; mediante el objeto creado, T-world se comunica con el programa agente.

Uso de un lenguaje distinto a JavaScript La diferencia está en el segundo paso de la creación del programa, ya que no se edita el programa dentro de T-World como se hizo en el caso de JavaScript. Como el usuario utilizará un lenguaje de programación externo al navegador web, se le debe indicar a T-World cómo enlazar el programa del usuario con su correspondiente programa agente. Para establecer el enlace, se ingresa la dirección IP y el puerto donde correrá el Proxy, y finalmente, un string que identifique al programa agente [1]. Este string, llamado *Magic String*, es utilizado por el Servidor Proxy para vincular el programa agente del usuario con su contraparte en T-World. En la próxima sección se explica con más detalle el uso del *Magic String*.
Notar que los programas agente escritos en lenguajes externos al navegador, tales como C o Java, presentan una diferencia significativa con los desarrollados en JavaScript en la manera en que se comunican con T-World, estos requieren del Servidor Proxy para comunicarse con T-World (ver figura 3b), por lo que finalmente el usuario deberá conectarse al proxy desde su programa agente, mediante un socket para recibir la percepción y devolver la respectiva acción a realizar por el agente. De este modo, el cuerpo del programa agente típicamente va a corresponderse con el siguiente algoritmo:

0) Crear socket
1) Conectar socket a la dirección IP y puerto del proxy
2) Enviar cadena de conexión
3) Recibir percepción
4) Según la percepción determinar la próxima acción.
5) Envia la acción
6) Si Percepción.Header distinto de END saltar a 3

3. El servidor Proxy de T-World

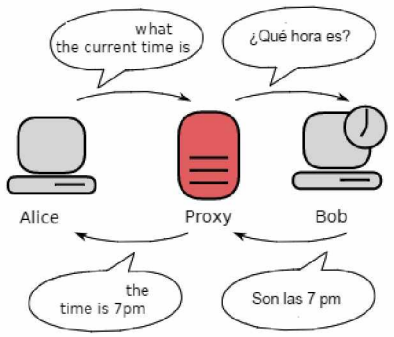
En esta sección se describen con más detalle dos aspectos importantes de la plataforma que, si bien se mencionaron antes, requieren una explicación más

[1] por defecto se le asigna el valor del nombre del programa agente

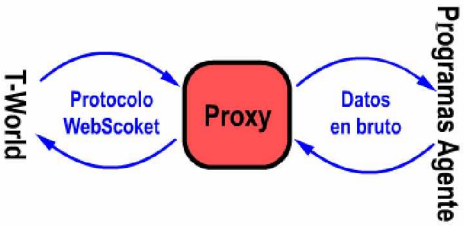
detallada: (a) el protocolo WebSocket que permite una comunicación bidireccional entre los programas agente con su correspondiente robot y (b) el formato estándar elegido para las percepciones: JSON (o XML), capaz de ser interpretado por programas escritos en distintos lenguajes.

Si bien tanto el protocolo como el formato elegidos son muy conocidos, requieren que el estudiante deba aprender a implementar WebSocket para comunicar el programa agente con el ambiente y conocer JSON para interpretar lo que sucede en el entorno. Esto complica la tarea del alumno al incorporar su programa agente, recibir las percepciones, y enviar las acciones a T-World. Esa complicación se solucionó con la creación de un *proxy*.

Un *servidor proxy* es un programa que rompe la conexión directa entre un emisor y un receptor, funcionando como un intermediario. Ambos, emisor y receptor piensan que se están comunicando entre sí, pero en realidad, lo hacen sólo con el proxy. Por eso su nombre, ya que proxy en inglés significa "actuar en nombre de otro". Este concepto se ilustra en la Figura 4a, en donde se ve



4 a)Idea general de un proxy



4 b)Proxy de T-World

la comunicación entre dos computadoras (Alice y Bob) conectadas a través de una tercera computadora que actúa como un Proxy. Notar que Bob desconoce que Alice hable Inglés y Alice que Bob habla Español, pero aún así pueden comunicarse normalmente ya que el Proxy se encarga de hacerlo transparente para ellos.

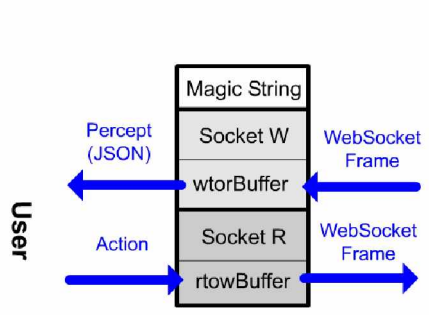
El proxy de la plataforma sigue el esquema planteado en la Figura 4a, donde Bob se corresponde con el programa agente, Alice con T-World, el idioma Inglés con WebSocket y el Español con las percepciones y las acciones.

El proxy se encarga de enlazar cada agente (robot) de la simulación 3D con su respectivo programa agente. Dado que la simulación transcurre dentro del navegador web, T-World asigna un WebSocket a cada robot y conecta cada WebSocket a su respectivo proxy. Por eso el diálogo entre T-World y el proxy sigue el protocolo WebSocket. A su vez, el programa del usuario sólo debe conectarse al proxy, recibir las percepciones JSON desde T-World y enviar las correspondientes acciones. Este proceso se ilustra en la figura 4b. El servidor

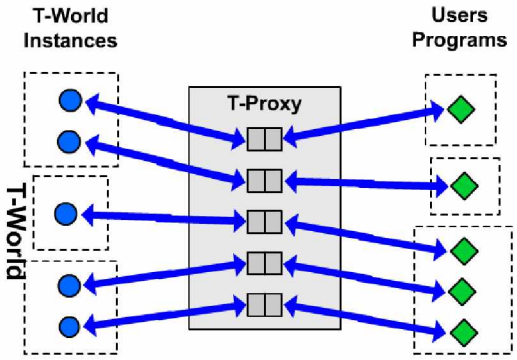
proxy ha sido codificado en lenguaje C, haciendo uso sólo de socket estándares (POSIX). Esto asegura la portabilidad a todas las plataformas y sistemas operativos. Para poder realizar su labor, el proxy dispone de un socket principal cuya tarea es la de escuchar nuevos pedidos de conexión, cada vez que uno de éstos arriba al servidor, se crea una nueva conexión. Esto último se lleva a cabo creando un nuevo socket por cada nueva conexión. Cuando uno de estos sockets reciben el mensaje correspondiente al handshake del protocolo WebSocket (ver sección 1.3 Opening Handshake de [4]), el proxy identifica a este socket como un WebSocket proveniente de T-World. Luego ambas partes, tanto los programas de usuario como los WebSockets envían un mensaje especial, llamado CONNECT, para que el proxy pueda identificarlos y aparearlos de forma apropiada. El mensaje CONNECT se forma por la cadena "CONNECT:" concatenada con la Magic String que el usuario le asignó al programa agente durante la creación del mismo en T-World. La Magic String es una cadena dada por el usuario que sirve como el identificador del programa agente, de forma tal que éste se enlace sólo con el robot correcto –aquel que tenga asignado la misma Magic String que su programa agente. Este Magic String se utiliza para armar pares de conexiones (WebSocket, Socket) bajo un mismo valor de Magic String. Estos pares se arman de la siguiente manera: cuando un mensaje CONNECT arriba desde un socket, a este socket se le asigna una etiqueta con el valor de la Magic String recibida. Posteriormente cuando un WebSocket envía el mensaje CONNECT se busca un socket libre[2] con la misma Magic String, de encontrarse uno se arma el par (WebSocket, Socket) con ellos. Este proceso también puede ocurrir a la inversa, que el WebScoket sea el primero en enviar el mensaje CONNECT y que posteriormente un socket lo haga. En este caso se busca un socket libre para este WebSocket. Para poder llevar a cabo el proceso antes descrito, el proxy genera una *estructura* por cada par de conexión. Esta estructura se ilustra en la figura 5a. Como se observa en la figura, esta estructura tiene un campo para almacenar la Magic String bajo la cual están apareados los sockets. También hay dos campos para guardar el socket de usuario (Socket R) y el WebSocket (Socket W) del par[3]. Los campos wtorBuffer y rtowBuffer son buffers de 16KB utilizados para el reenvío de mensajes desde el WebSocket hacia el usuario y viceversa, proceso que se describe en las siguientes secciones.

La estructura descripta le brinda al proxy la capacidad de soportar múltiples pares de conexiones, todos conectados de manera concurrente al proxy. Como se observa en la figura 5b, cada estructura se encarga de mantener un par de conexión entre múltiples instancias de T-World y múltiples instancias de programas de usuarios. Los círculos representan WebSockets, y el cuadrado donde éstos se encuentran instancias de T-World. Los rombos representan sockets de usuarios, cada cuadrado contenedor representa una instancia de programa de usuario.

[2] por libre se refiere a que no pertenezca ya a un par (WebSocket, Socket)
[3] Socket R y Socket W guardan el descriptor de archivo de cada socket



5 a) Interacción Proxy-T-World



5 b) Interacción Agente-Proxy

3.1. Interacción entre Proxy y T-World

Como se mencionó anteriormente la comunicación entre el proxy y T-World se hace por medio del protocolo WebSocket. Esto implicó implementar manualmente dicho protocolo durante el desarrollo del proxy, siguiendo las especificaciones detalladas en su respectivo RFC [4]. Los mensajes que se envían utilizando el protocolo WebSocket tienen un formato específico el cual se denomina como "WebSocket Frame" o en este contexto simplemente Frame. Los datos que viajan dentro de Frames que se envían desde T-World hacia el proxy lo hacen enmascarados –esto es uno de los requerimientos del protocolo. Por esta razón cuando se recibe un Frame desde T-World el proxy debe desenmascarar el dato enviado (la percepción) y almacenarlo en el buffer wtorBuffer para ser posteriormente reenviado hacia su correspondiente programa agente (ver figura 5a). Posteriormente cuando el programa de usuario responde con la acción a realizar por el robot, esta se guarda en el buffer rtowBuffer. El proxy obtiene dicha acción desde este buffer, la encapsula dentro de un WebSocket Frame y finalmente envía este Frame al WebSocket correspondiente dentro de T-World.

3.2. Interacción entre programa agente y Proxy

La interacción en este caso es mucho más simple, ya que no se realiza utilizando ningún protocolo en particular. La percepción almacenada en el buffer wtorBuffer se envía hacia el programa agente sin modificarla. Esto es, la secuencia de caracteres que se encuentran en el buffer se envían hacia el programa agente. Como esta secuencia de caracteres tiene un formato especial, JSON, puede ser interpretada por el programa agente receptor. Una vez interpretada, el programa agente típicamente realizará algún cómputo sobre la misma y posteriormente devolverá una acción. Las acciones son también secuencias de caracteres. Una vez recibida la acción, el proxy la encapsula dentro de un WebSocket Frame y escribe los bytes del Frame en el buffer rtowBuffer. Acto seguido este Frame se envía hacia el correspondiente WebSocket –esto provoca que el robot correspondiente actúe dentro la simulación 3D.

4. Conclusiones y Consideraciones finales

Se describe un entorno para evaluar el desempeño de agentes y particularmente se detalla la implementación del servidor proxy creado especialmente para integrar agentes escritos en diferentes lenguajes. En procura de crear un entorno portable, se eligió como lenguaje de implementación JavaScript y así poder ejecutar en el ámbito de un navegador web. Los usuarios que también elijan Java Script para programar sus agentes, tendrán una API y una librería de agentes a su disposición. Otra cualidad destacada es la flexibilidad porque los agentes se pueden programar en distintos lenguajes como C, Java o Prolog. Mediante un servidor Proxy, distintos agentes heterogéneos se integran a T-World. El proxy usa WebSocket para hablar con el robot de T-World y transmitir información en formato JSON a los programas agente que controlan al robot. La conexión entre el robot y su controlador se mantiene en el proxy mediante una estructura de datos que contiene dos buffers para la comunicación bidireccional.

Los objetivos que se plantearon desde el inicio del diseño de T-World (tworld-ai.com), fueron variados, entre otros que fuera accesible desde cualquier parte del mundo, de manera que simplemente ingresando a un sitio web se tenga acceso a una herramienta completa e integrada de forma nativa al navegador web. Que su funcionamiento fuese independiente de un sistema operativo en particular o de cualquier otra pieza de software –como intérpretes, runtime frameworks o máquinas virtuales. Para lograr todos estos objetivos, entre otras cosas, fue requerido codificar más de 18 mil líneas de código –12 mil en JavaScript, aproximadamente mil en Lenguaje C y el resto en HTML y PHP. También fue nacesaria la utilización de diversas tecnologías tales como AngularJS de Google como framework para la aplicación web y la librería CopperLitch como una abstracción de WebGL para la generación de gráficos 3D.

Referencias

1. Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE.* John Wiley & Sons, NJ, April 2007.
2. Sergio Burdisso, Guillermo Aguirre, and Marcelo Errecalde. T-world: un entorno gráfico, flexible y portable para la enseñanza e investigación de agentes inteligentes. In *CACiC 2014.* Universidad Nacional de La Matanza, 2014.
3. S. Hanks, M. Pollack, and P. Cohen. Benchmarks, test beds, controlled experimentation and the design of agent architectures. *AIMagazine*, 14(4):17–42, 1993.
4. I. Fette and A. Melnikov. The WebSocket Protocol. Technical report, RFC Editor, December 2011.
5. Martha E. Pollack and Marc Ringuette. Introducing the tileworld: Experimentally evaluating agent architectures. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 183–189, 1990.
6. Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.).* Pearson Education, 2010.