UNIVERSIDAD NACIONAL DE COLOMBIA

DESARROLLO DE UN SISTEMA MULTI-AGENTE PARA EL DIAGNÓSTICO DE FALLAS EN SERVICIOS DE INTERNET CORPORATIVO

ALFONSO AYALA PALOMA

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial

Bogotá, Colombia

2016

DESARROLLO DE UN SISTEMA MULTI-AGENTE PARA EL DIAGNÓSTICO DE FALLAS EN SERVICIOS DE INTERNET CORPORATIVO

ALFONSO AYALA PALOMÁ

Trabajo final presentado como requisito parcial para optar al título de:

Magister en Ingeniería de Sistemas y Computación

Director:
Luis Fernando Niño Vásquez, Ph.D.

Grupo de Investigación:
Laboratorio de Investigación en Sistemas Inteligentes (LISI)

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial
Bogotá, Colombia
2016

Dedicada a mis familias, aquella donde nací y aquellas que también me acogieron con los brazos abiertos.

Agradecimientos

Al profesor Luis Fernando Niño Vásquez, Director del Grupo de Investigación LISI, a los miembros de este grupo, a mis compañeros de trabajo y a mi familia, este trabajo no hubiera sido posible sin su colaboración, comprensión y paciente aprobación.

Resumen

En empresas de telecomunicaciones, varios dispositivos deben trabajar de forma coordinada para brindar un servicio a sus clientes cumpliendo expectativas de calidad y disponibilidad. Cuando ocurre una falla, el servicio debe restablecerse rápidamente, por medio de un diagnóstico y un conjunto de acciones correctivas. Este proyecto presenta una aplicación de los sistemas multi-agente para realizar diagnósticos de fallas en servicios de acceso a internet. El sistema cuenta con agentes diagnosticadores, de conexión a dispositivos, administradores de dispositivos y los encargados de la conexión a bases de datos. El sistema fue desarrollado usando la plataforma para sistemas multi-agente JADE. En las pruebas realizadas, el sistema fue capaz de establecer los diagnósticos correctos en los escenarios configurados para tal fin, de acuerdo con los expertos humanos consultados. El proyecto presenta una solución alternativa a las de los proveedores de dispositivos de telecomunicaciones, las que, generalmente, se restringen a una única marca y no ofrecen una presentación integrada del servicio.

Abstract

In telecommunication companies, several devices must work in a coordinated way to fulfill provide services to clients. When a failure occurs, the service must be quickly restored, by diagnosing the failure and taking pertinent corrective actions. This project shows a MAS solution to diagnose failures on internet access services. This system has a diagnosticator, a device connection and a database connection agents. The system was developed using the JADE platform. In the testing phase, the system was able to perform the right diagnostics on some test scenarios, which matched the human expert's opinions. This project presents an alternative to vendor solutions which are, generally, restricted to one brand and could not offer an integrated view for the service.

Contenido

1. Propuesta trabajo final 4
1.1 Objetivos 4
1.1.1 Objetivo General 4
1.1.2 Objetivos Específicos 4
1.2 Justificación 4
1.2.1 Pertinencia 4
1.2.2 Tipo de articulación del proyecto a políticas, planes o programas 5
1.3 Estado del arte 6
2. Marco teórico 8
2.1 Gestión de servicios de tecnología de la información 8
2.2 Librería ITIL 9
2.3 Operación del servicio 12
2.3.1 Gestión de incidentes 13
2.3.2 NOC – *Network Operation center* 17
2.3.3 Sistemas de gestión de incidentes 17
2.4 Agentes inteligentes 18
2.4.1 Agentes 18
2.4.2 Sistemas multi-agente 19
2.4.3 Clasificación de arquitecturas de sistemas multi-agente 20
2.4.4 Arquitecturas basadas en lógica 20
2.4.5 Arquitectura reactiva 21
2.4.6 Arquitectura BDI (*Belief-Desire-Intention*) 21
2.4.7 Arquitectura en capas (híbrida) 22
2.4.8 Arquitectura cognitiva 23
2.4.9 Estructura de un sistema multi-agente 24

2.4.10 Arquitectura estándar de la FIPA 25
3. Análisis de incidentes más frecuentes en servicios de acceso a internet corporativo 29
3.1 Motivación 29
3.2 Obtención de datos 29
3.3 Análisis descriptivo de los datos 31
3.3.1 Cantidad de incidentes diarios 31
3.3.2 Tiempo de solución de incidentes 33
3.3.3 Categorías de clasificación de incidentes 34
3.4 Diagnósticos a realizar 36
3.4.1 Selección de diagnósticos 36
3.4.2 Proceso actual de diagnóstico 38
4. Requerimientos del sistema multi-agente a desarrollar 42
4.1 Perspectiva del sistema 42
4.2 Funciones del sistema 43
4.3 Clases de usuarios y características 44
4.4 Ambiente de operación 44
4.5 Restricciones de diseño e implementación 45
4.6 Supuestos y dependencias 45
4.7 Características del sistema 46
5. Análisis y diseño del sistema multi-agente 47
5.1 Selección de la herramienta y metodología 47
5.2 Metodología y herramientas empleadas 48
5.3 Planeación 50
5.4 Análisis 50
5.4.1 Casos de Uso 51
5.4.2 Identificación inicial de los tipos de agentes 52
5.4.3 Identificación de responsabilidades 52
5.4.4 Identificación de las interacciones 53

5.4.5 Refinamiento de agentes 55
5.4.6 Información de despliegue de agentes 57
5.5 Diseño 58
5.5.1 División o unión de agentes 58
5.5.2 Especificación de interacción 58
5.5.3 Definición del protocolo de interacción ad-hoc 60
5.5.4 Estructura de mensajes 60
5.5.5 Páginas amarillas de agentes 62
5.5.6 Interacciones agente-recurso 62
5.5.7 Interacciones agente-usuario 63
5.5.8 Comportamientos internos del agente 63
5.5.9 Definición de una ontología 64
5.5.10 Selección del lenguaje de contenido 67
6. Implementación 69
6.1 Herramientas usadas 69
6.2 Descripción de la solución 69
6.3 Modelado de las soluciones brindadas a los incidentes 75
6.4 Software desarrollado 76
6.5 Implementación del sistema usando principios de una metodología ágil 77
6.6 Uso de la herramienta JADE 78
7. Pruebas 83
7.1 Definición de pruebas 83
7.2 Definición de red de dispositivos de pruebas 83
7.3 Escenarios de prueba 85
7.3.1 Escenario de servicio sin falla. 86
7.3.2 Escenario de CPE bloqueado 89
7.3.3 Escenario de CPE con error en su configuración 91
7.3.4 Escenario de falla por corte de fibra 93
7.3.5 Escenario de falla en *router* PE 95

7.3.6 Escenario de falla por tabla ARP sin registrar dispositivos en *switch* 97
7.3.7 Escenario de falla por interfaz *switch* apagada..... 99
7.3.8 Escenario de falla por *switch* apagado 101
8. Conclusiones y recomendaciones 104
8.1 Conclusiones.....104
8.2 Recomendaciones.....107
Anexo 1. Programa para ejecutar comandos.....110
Bibliografía 113

Lista de figuras

Figura 1. Ciclo de vida del servicio ITIL. Tomado de Von Bon, J et al., [44]..... 10
Figura 2. Etapas del ciclo de vida del servicio ITIL. Tomado de Von Bon, J et al., [44]... 11
Figura 3. Proceso de gestión de incidentes. Tomado de Tomado de Von Bon, J et al., [44] 16
Figura 4. Estructura de un sistema multi-agente. Tomada de Millet, S. et al., [58]..... 24
Figura 5. Estructura estándar para sistemas multi-agente FIPA. Tomada de Poslad, S. [59]. 26
Figura 6. Fragmento de los datos analizados. Tomado de la empresa de telecomunicaciones..... 30
Figura 7. Ficha de la muestra tomada. Creada por el autor. 31
Figura 8. Serie de tiempo de llegadas de incidentes. Creada por el autor usando R a partir de los datos de la empresa de telecomunicaciones..... 32
Figura 9. Descomposición de la serie de tiempo de llegadas de incidentes. Creada por el autor en R. a partir de los datos de la empresa de telecomunicaciones 33
Figura 10. Distribución de los tiempos de atención de incidentes. Creada por el autor en R. a partir de los datos de la empresa de telecomunicaciones 34

Figura 11. Clasificación de incidentes encontrada. Creada por el autor a partir de los datos de la empresa de telecomunicaciones 35
Figura 12. Distribución de tipos de incidentes. Los números indican la cantidad de incidentes por cada tipo. Tomado de la empresa de telecomunicaciones..... 36
Figura 13. Causas raíz con cantidad de incidentes y porcentajes del total. Tomado de la empresa de telecomunicaciones. 37
Figura 14. Lista de diagnósticos. Tomado de la empresa de telecomunicaciones. 37
Figura 15. Diagrama de un servicio de acceso a internet. Tomado de la empresa de telecomunicaciones. 39
Figura 16. Tabla de comandos a ejecutar en los dispositivos. Creado por el autor a partir de la información suministrada por la empresa de telecomunicaciones..... 39
Figura 17. Diagnósticos a realizar por el sistema. Creado por el autor a partir de la información suministrada por la empresa de telecomunicaciones 40
Figura 18. Reglas para encontrar los diagnósticos. Creado por el autor a partir de la información suministrada por la empresa de telecomunicaciones 41
Figura 19. Perspectiva del sistema. Creado por el autor. 43
Figura 20. Metodología de análisis y diseño de MAS. Tomado de Nikraz, M. et al. [66]. 49
Figura 21. Casos de uso del sistema. Creado por el autor 51
Figura 22. Identificación de agentes. Creado por el autor. 52
Figura 23. Diagrama de agentes con sus interacciones. Creado por el autor. 53
Figura 24. Diagrama de agentes luego del paso de refinamiento. 55
Figura 25. Despliegue de agentes. 57
Figura 26. Conceptos en la ontología. 65
Figura 27. Predicados de la ontología. 66
Figura 28. Acciones de los agentes. 66
Figura 29. Ontología de diagnóstico de incidentes. 67
Figura 30. Clases del sistema. 70
Figura 31. Proceso de diagnóstico. 71
Figura 32. Paquetes del sistema. 72
Figura 33. Hardware usado en el sistema. 73
Figura 34. Casos de uso del sistema. 74
Figura 35. Representación de un sistema de producción de reglas. Tomado de Drools Team, [72]. 75
Figura 36. Diagrama del software desarrollado. (Creado por el autor). 76
Figura 37. Interfaz de JADE. 79
Figura 38. Agentes en el contenedor. 79
Figura 39. Envío de mensajes en JADE. 80
Figura 40. Proyecto en eclipse. 81
Figura 41. Conexión de la aplicación web. 81

Figura 42. Red de pruebas. 84
Figura 43. Equipos a utilizar en las pruebas. 84
Figura 44. Escenarios de prueba. 85
Figura 45. Parte de la configuración de un dispositivo de pruebas. 85
Figura 46. Resultado para el escenario sin falla. 87
Figura 47. Respuesta para el escenario de CPE bloqueado. 90
Figura 48. Respuesta para el escenario de CPE con error en su configuración. 92
Figura 49. Respuesta para el escenario falla por corte de fibra. 94
Figura 50. Respuesta para el escenario de falla en *router* PE. 96
Figura 51. Respuesta escenario sin dispositivos a nivel ARP en *switch*. 98
Figura 52. Respuesta para escenario de interfaz de switch apagada. 100
Figura 53. Respuesta para escenario de switch apagado. 102

Lista de tablas

Tabla 1. Tipos de mensajes en FIPA. Tomada de Poslad, S. [59]. 27
Tabla 2. Identificación de responsabilidades de los agentes. 53
Tabla 3. Tabla de responsabilidades luego de identificar las interacciones entre agentes. 54
Tabla 4. Agentes y responsabilidades, después del paso de refinamiento. 57
Tabla 5. Especificación de interacciones entre agentes. 59
Tabla 6. Estructura de los mensajes..... 62

Lista de abreviaturas

Abreviaturas

Abreviatura	Término
MAS	*Multi-agent System* (Sistema Multi-Agente)
JADE	*Java Agent DEvelopment Framework*
PE	*Provider Edge Router*
CPE	*Customer Premises Equipment*
MPLS	*Multiprotocol Label Switching*

Introducción

Los proveedores de dispositivos de comunicaciones como Cisco, Huawei y Arris suministran software a las empresas prestadoras de servicios de telecomunicaciones para administrar sus plataformas. Para que un ingeniero de atención al cliente pueda brindar una solución rápida y acertada cuando se presenta una falla en un servicio, se requiere que haya sido capacitado, que cuente con experiencia en el uso del software de administración y en la solución de los incidentes. Generalmente, cada sistema administra únicamente una plataforma y no se conecta con otras, dificultando tener una visión completa del servicio.

Estos sistemas no poseen la funcionalidad de definir reglas ni acciones auto-correctivas y muchos de ellos son únicamente de monitoreo, tal como la aplicaciones Solarwinds [1] la cual se concentra en dispositivos de red como routers, Cacti [2] herramienta de monitoreo vía protocolo snmp (*System Network Management Protocol*) con almacenamiento en archivos RRD (*Round Robin Data*) y Cheetah [3].herramienta de monitoreo para equipos en redes HFC (*Hybrid Fibre-Coaxial*).

Al presentarse una falla, según la documentación manejada por el NOC (del inglés, *Network Operation Center*) de primer nivel, para obtener un diagnóstico, se deben ejecutar varios comandos en los dispositivos distribuidos geográficamente que componen el servicio. Partiendo de este diagnóstico se pasa a la etapa de actuación en la que se ejecutan comandos correctivos en cada dispositivo que lo requiera.

Estos dos pasos de diagnóstico y actuación se repiten hasta que el problema quede resuelto o hasta que sea necesario escalar a otro nivel. Esta labor se realiza manualmente siguiendo la documentación y apoyada en la experiencia del ingeniero que

esté brindando la solución. Esta tarea es altamente manual y presenta una oportunidad de mejora.

Entonces, la empresa presenta la necesidad de automatizar el proceso de diagnóstico para disminuir los tiempos de atención de fallas y disminuir los errores humanos en la interacción con diversas plataformas.

Para satisfacer esta necesidad y debido a que los equipos que hacen parte de un servicio son heterogéneos en cuanto a capacidades y tecnología, que están distribuidos geográficamente y están conectados a la misma red, un sistema multi-agente podría desarrollarse para resolver el problema mencionado.

Los sistemas multi-agente han sido aplicados en múltiples áreas. Como trabajo del Laboratorio de Investigación en Sistemas Inteligentes (LISI) de la Universidad Nacional de Colombia se pueden resaltar las tesis de maestría de Ardila [4], en la cual se desarrolló una arquitectura para el manejo de congestión de redes de datos; la de Castro [5], donde se analizó el tráfico de redes de área local y la de Sánchez [6], en la cual se propuso un modelo para tratar la congestión en redes de datos. En estos trabajos se tomaron datos de la red y se hizo un diagnóstico de la situación actual de la red y un tratamiento para la situación diagnosticada.

En el presente trabajo se propone un enfoque basado en un sistema multi-agente para realizar diagnósticos de incidentes sobre servicios de internet corporativo. El sistema utiliza agentes para realizar las tareas de ejecutar comandos en los dispositivos y llegar al diagnóstico usando un conjunto de reglas y los resultados de la ejecución de comandos.

Para obtener información sobre incidentes típicos y sus diagnósticos, se toma información real de una empresa de telecomunicaciones que presta servicios de acceso a internet, su nombre no se divulga por aspectos de confidencialidad de la información.

Este documento se estructura en siete capítulos, el primero con el marco teórico que comprende manejo de incidentes y sistemas multi-agente, el segundo muestra la revisión de los incidentes y especificación de los diagnósticos a determinar, el tercero presenta los requerimientos del sistema, el cuarto el análisis y diseño, el quinto, la implementación, en el sexto las pruebas y en el séptimo, conclusiones y recomendaciones.

1. Propuesta trabajo final

1.1 Objetivos

1.1.1 Objetivo General

Desarrollar un sistema multi-agente para el diagnóstico de incidentes en servicios de internet corporativo en empresas de telecomunicaciones.

1.1.2 Objetivos Específicos

Determinar y analizar los incidentes más frecuentes y las soluciones brindadas por los funcionarios de empresas de comunicaciones en incidentes de servicios de internet corporativo.

Modelar las soluciones brindadas a los incidentes en servicios de internet corporativo en empresas de telecomunicaciones.

Diseñar e implementar un sistema multi-agente para diagnosticar los incidentes en servicios de internet corporativo en empresas de telecomunicaciones.

1.2 Justificación

1.2.1 Pertinencia

Las empresas de telecomunicaciones comparten un mercado competido, en el cual los clientes son sensibles a los precios y tienen un poder de negociación medio. La tasa de clientes que cancelan el servicio, es una de sus principales preocupaciones, además del fenómeno de migración e incluso rotación de clientes de una empresa a otra. Uno de los aspectos determinantes en la permanencia de un cliente es la rápida solución de fallas. Si una empresa puede satisfacer esta necesidad, es posible que alcance una ventaja competitiva y por lo tanto pueda estar en una mejor posición estratégica. Adicionalmente,

el tamaño de operación de soporte actual y su costo, supera al de un sistema automático que realice las mismas funciones, por esto, el desarrollo de este proyecto es pertinente.

1.2.2 Tipo de articulación del proyecto a políticas, planes o programas

Las empresas suscriben acuerdos de nivel de servicio (SLAs, Service Level Agreement, por su sigla en inglés) con sus proveedores de tecnología entre ellos, las empresas de telecomunicaciones. Estos acuerdos pueden incluir cláusulas pecuniarias en caso de tiempos fuera de servicio o incumplimiento de valores como ancho de banda de subida (upstream) o bajada (downstream). De esta forma, las empresas de telecomunicaciones deben garantizar una disponibilidad pactada a sus clientes. Para ello, es necesario resolver los incidentes normales que impliquen afectación de servicio en el menor tiempo posible de tal forma que se cumplan los acuerdos de nivel de servicio. Entonces este proyecto se articula con las iniciativas de ahorro y automatización de procesos en las empresas de telecomunicaciones.

El interés personal consiste en aplicar la ingeniería de sistemas en la solución de problemas reales de Colombia, específicamente en la empresa para la cual labora el autor. Este proyecto se encuentra en la categoría de optimización de recursos.

Las técnicas existentes de ejecutar comandos manualmente están sujetas a errores humanos y dependen tanto de la experiencia como del conocimiento de las personas que las ejecutan. En la medida que el número de servicios activos crezca, la cantidad de ingenieros dedicados a solucionar incidentes también debería crecer, sin embargo, su labor “mecánica” en cuanto a seguir los procedimientos, dejaría a las empresas en una deficiente posición competitiva, cuando la competencia utilice medios automatizados más ágiles para solucionar sus incidentes.

Las empresas de telecomunicaciones buscan reducir sus costos operativos, sin afectar la calidad de sus servicios. En la actualidad, el tamaño de un Centro de Operaciones de Red (NOC), encargado de la solución de incidentes es del orden de 200 personas en empresas como Claro y Movistar. Un NOC debe estar compuesto por ingenieros con conocimientos y experiencia en las diferentes arquitecturas de soluciones que ofrezca la empresa, lo cual cobija tecnologías como MPLS, GPON y SDH, entre otras. Un error humano en estas plataformas o una demora en solucionar un incidente puede significar pérdidas económicas por incumplimiento de acuerdos de niveles de servicio o por pérdida de clientes.

1.3 Estado del arte

Según Ghosh, D. et al. [7], los agentes de un MAS pueden actuar autónomamente y proveer capacidades de auto-reparación, que protegen la parte del sistema asignada. Otros enfoques consideran la colaboración de capacidades de agentes.

Los agentes han sido utilizados en la solución de problemas de diagnóstico y tolerancia a fallas, en este aspecto se destacan los trabajos de Corsava y Getov [8], quienes presentaron agentes que diagnostican comportamiento de diferentes clases de recursos (hardware, sistemas operativos y recursos de red). Estos poseen monitoreo, diagnóstico, reparación, colaboración.

Otros autores como Huhms et al. [9] trabajan con la redundancia, la cual cobija redundancia por diversidad (*redundancy by diversity*) para lograr diversidad de implementación y redundancia por votación (*redundancy by voting*) para lograr un resultado correcto.

Un enfoque centralizado puede ser encontrado en la arquitectura de software Unity, descrito en Tesauro et al [10]. APLE de IBM es un framework basado en agentes de Java, construido sobre la idea de agentes cuya funcionalidad puede ser modificada para convertirlos en nuevos agentes que manejen situaciones diferentes.

En cuanto a las metodologías de desarrollo de MAS reseñadas por Ardila, E. [4], se encuentran: GAIA descrita por Gómez, S. et al [11], Tropos descrita en [12], JADE presentada en [13], O-MaSE descrita por Deloach, S. et al [14], y CoMoMas, Agent UML, MasCommonKADS, presentadas en Iglesias, C. et al. [15].

Según Wooldridge et al. [16], Los lenguajes de comunicación y contenido entre agentes más utilizados son: KQML y KIF, descritos en [17], FIPA-ACL y FIPA-SL, presentados en [18].

Las plataformas para el desarrollo de sistemas multi-agente comparadas por Sánchez, J. en [6] son: Aglets descrita en [19], JADE en [13], JACK en [20], Grasshoper en [21], FIPA-OS en [22], AgentTools [11], y Zeus en [23].

En la revisión de MAS en la industria y los negocios, según Jarvis, J. et al. [24], se encuentran aplicaciones en: Planeación presentadas por Colombo, A. et al [25], programación presentada por Zhang, et al [26], diagnósticos por Davidson, E. et al [27], monitoreo de condiciones presentadas por McArthur, S. et al. [28], y Buse, D. et al. [29], Control distribuido, presentadas por Zhou, J. et al. [30] y Galdun, J. et al. [31] , control híbrido por Fregene, K. et al. [32], control de congestión, presentados por Hwang, K. et al en [33] y Srinivasan et al. [34], restauración de sistemas presentada por Nagata, T. et al. [35], Simulación de mercados presentada por Widergren, S. et al. [36] y Koesrindartoto, D.

[37], control de redes presentada por Dimeas, A. et al. [38] y Hwang, K. et al. [33] y automatización presentada por Srinivasan, D. et al. [34].

Adicionalmente los trabajos en juegos de guerra (*War gaming*) presentados por Heinze et al. [39] y administración de la misión (*mission management*) presentado por Karim et al. [40], muestran la versatilidad y aplicabilidad de los MAS para la solución de problemas muy diversos.

2. Marco teórico

2.1 Gestión de servicios de tecnología de la información

Según de Pablos, C. et al. [41], la gestión de servicios de tecnología de la información (ITSM, por su sigla en inglés), se considera una tarea fundamental y estratégica en las organizaciones. El interés en ITSM se ha motivado no solo por la competencia entre las compañías, sino también por el interés de las organizaciones en integrar la tecnología de la información (IT) a la estrategia competitiva.

Un proceso adecuado de ITSM permitirá a las compañías

- mejorar continuamente sus procesos,
- alcanzar una mejora significativa en la calidad percibida por los clientes y
- mejorar su estrategia de IT.

La información es uno de los activos más importantes de todas las organizaciones. Así los servicios de IT son decisivos para la correcta gestión del conocimiento, toma de decisiones y planeación de acciones para la compañía [41].

Los servicios de IT están generando un cambio cultural en las organizaciones debido a que permiten integrar los sistemas de información al negocio [41] . Los servicios de IT se requieren para evolucionar y rápidamente adaptarse a las nuevas necesidades y tecnologías de las compañías, aún más teniendo en cuenta la proliferación de servicios compartidos de IT y la externalización (*outsourcing*).

Sin embargo, la oportunidad estratégica que las nuevas tecnologías traen a las compañías es simultáneamente la causa de las dificultades que aparecen en su gestión. Por ello, se han desarrollado marcos de referencia (*frameworks*) como ITIL, eTOM, COBIT, que reúnan las mejores prácticas y recomendaciones para enfrentar estos retos [41].

Las mejores prácticas documentadas en varios marcos de referencia ITSM han motivado a muchas compañías a pasar de una organización orientada a productos a una orientada a servicios. El enfoque basado en procesos ha sido considerado como una forma más

efectiva de alcanzar esta integración de IT a la estrategia de la organización [41]. Este enfoque se utiliza en uno de los marcos de referencia más usado en empresas de IT, ITIL, el cual se describe a continuación.

2.2 Librería ITIL

La librería ITIL (*Information Technology Infrastructure Library*) es un conjunto de mejores prácticas representadas en procesos para la administración de servicios de tecnología de la información. ITIL incluye desde procesos de niveles de servicio y administración de la capacidad hasta configuración, liberación y control de cambios [42].

ITIL es el estándar de facto generalmente reconocido para los procesos de gestión de sistemas de IT. El principal objetivo de la librería ITIL es "establecer las mejores prácticas y un estándar de calidad de servicio de IT que los clientes deberían exigir y los proveedores deberían buscar satisfacer". ITIL fue desarrollado originalmente por el gobierno británico a través de la Agencia Central de Computadores y Telecomunicaciones (CCTA). Ahora ITIL es mantenida por el *Netherlands IT Examinations Institute* (EXIN) [43].

Un servicio se define como una forma de entregar valor a los clientes por medio de los resultados que ellos quieren lograr, sin asumir los costos y riesgos que implique el mantener los recursos necesarios para prestarlos [44]. Ejemplos de servicios son telefonía fija, telefonía móvil, televisión, internet y e-mail.

En ITIL, el ciclo de vida del servicio se divide en estrategia, diseño, transición, operación y mejoramiento continuo. Una representación gráfica se observa en la figura 1.

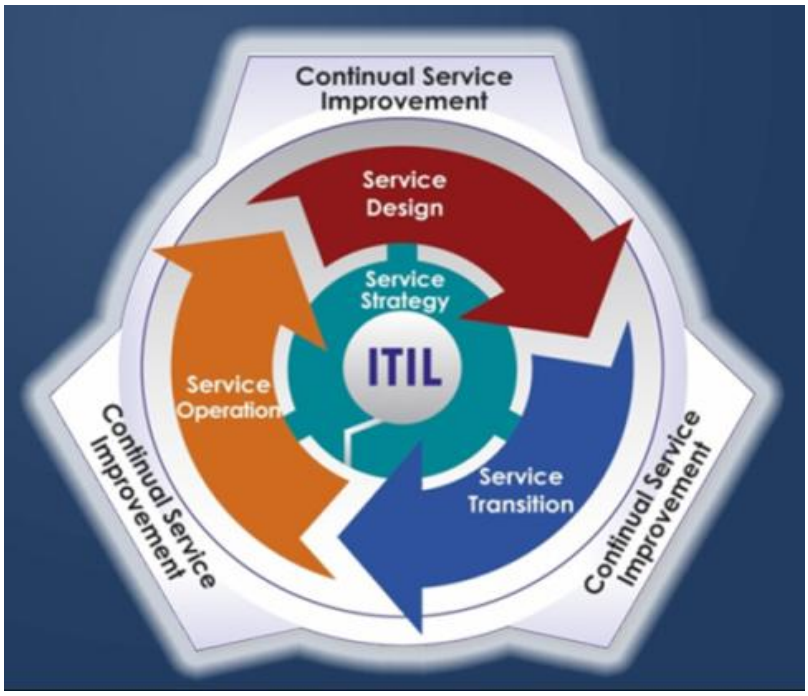

Figura 1. Ciclo de vida del servicio ITIL. Tomado de Von Bon, J et al., **[44]**

Cada etapa del ciclo de vida tiene asociados procesos, gráficamente se pueden observar en la figura 2.

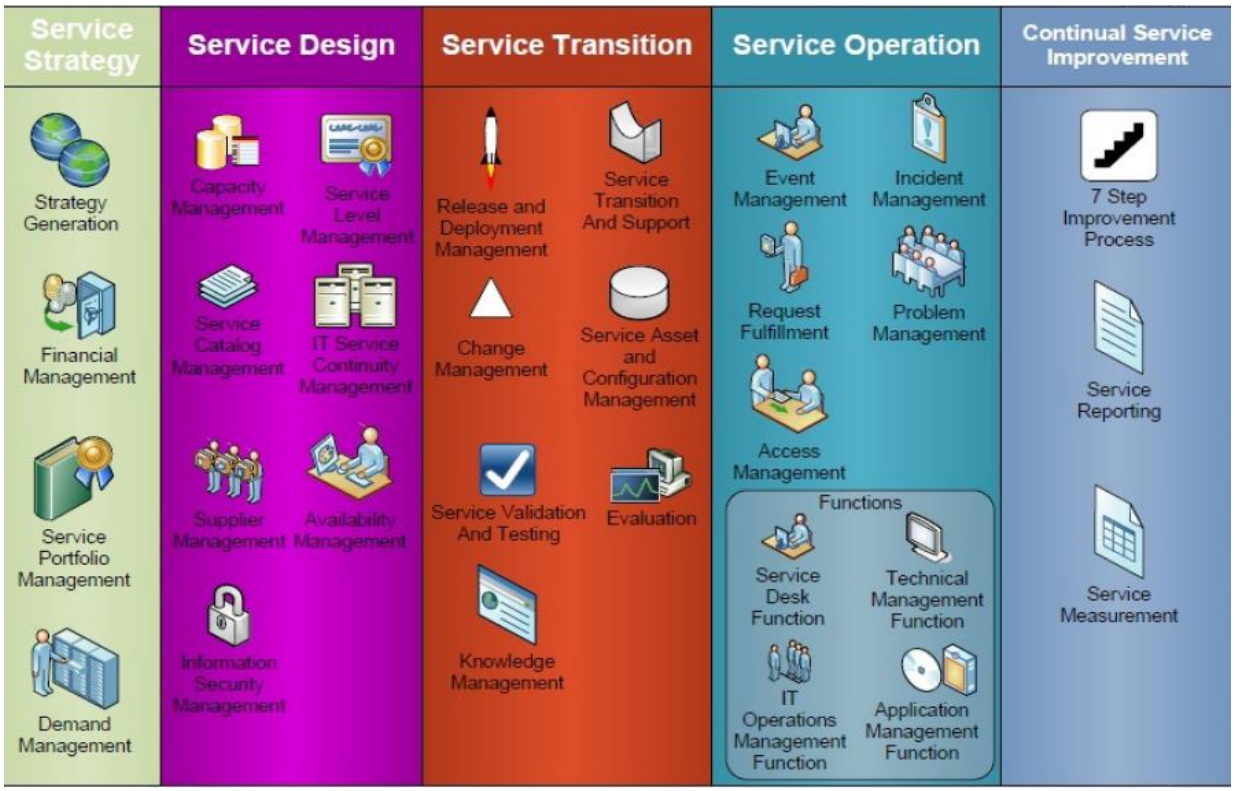

Figura 2. Etapas del ciclo de vida del servicio ITIL. Tomado de Von Bon, J et al., **[44]**.

Estrategia del servicio (*Service Strategy*), se enfoca en la identificación de las oportunidades del mercado para las cuales se pueden desarrollar requerimientos de clientes internos o externos. De otro lado, puede ser vista como una revisión de los objetivos y expectativas generales de la organización, asegurando que la estrategia de IT esté alineada con ellos [45].

Diseño del servicio (*Service Design*), inicia con un conjunto de requerimientos de negocio, ya sean nuevos o modificados, y termina con una solución diseñada para cumplir las necesidades documentadas de la organización [45].

Transición del servicio (*Service Transition*), trata la gestión de cambios, riesgos y aseguramiento de la calidad durante el despliegue de los diseños del servicio de tal forma que las operaciones del servicio puedan gestionar los servicios y soportar la infraestructura de una forma controlada [45].

Operación del servicio (*Service Operation*), trata de actividades cotidianas para mantener los servicios operativos una vez se ponen en producción [45].

Mejoramiento continuo del servicio (*Continual Service Improvement*), provee una visión general de todos los elementos y busca la manera de mejorar el proceso global y la provisión de servicios [45].

La librería ITIL describe las prácticas en términos de planeación, implementación, auditoría, beneficios, costo y posibles problemas y soporte de herramientas. Presenta, además, procedimientos operacionales, roles, responsabilidades, dependencias, procesos que soportan y entrenamiento necesarios para que la organización pueda seguir estas prácticas [44].

El uso de ITIL en lugar de otros marcos de trabajo se justifica por su buena reputación en el mercado. ITIL ha sido probado y usado mundialmente. Debido a que es flexible y escalable, organizaciones de todos los tamaños pueden implementar partes de ITIL para obtener beneficios. Además, al adoptar este estándar, las empresas pueden ganar ventaja competitiva a través de la creación de valor, cambio ágil, así como, incrementar la satisfacción del cliente gracias a su fiabilidad [44].

A continuación se presentará la etapa del ciclo de vida del servicio en la que se enmarca el presente trabajo.

2.3 Operación del servicio

El principal objetivo de operación del servicio es coordinar y llevar al cabo las actividades y procesos requeridos para entregar y gestionar servicios en los niveles acordados a los

usuarios del negocio y clientes. Operación del servicio es responsable de la gestión de la tecnología que se usa para entregar y soportar servicios [45].

Los procesos bien diseñados y bien implementados tendrían muy poco valor si la operación diaria de estos procesos no se conduce, controla y administra apropiadamente, ni tampoco las mejoras en el servicio serían posibles si las actividades diarias para monitorear el desempeño, calcular métricas y reunir datos no son conducidas sistemáticamente durante la operación del servicio [45].

La operación del servicio cuenta con los procesos de gestión de eventos, gestión de incidentes, cumplimiento de peticiones, gestión de problemas y gestión de acceso [44]. El presente trabajo está enmarcado en el proceso de gestión de incidentes.

2.3.1 Gestión de incidentes

El objetivo principal de la gestión de incidentes es restaurar la operación normal del servicio tan pronto como sea posible y minimizar el efecto adverso sobre las operaciones de negocio. Se busca, además, mantener óptimos niveles de servicio, calidad y disponibilidad [42].

En el contexto de gestión de incidentes, se definen los siguientes conceptos:

- Incidente: cualquier evento que no forma parte de la operación normal de un servicio y el cual causa o puede causar una interrupción o una reducción de la calidad de ese servicio [45] .
- Problema: causa de uno o más incidentes [45].
- Solución alternativa (w*orkaround)* en un medio temporal para resolver problemas en servicios de TI [45].
- Error conocido: problema documentado del cual se conoce su causa raíz (*root cause*) y un *workaround*. [45]

El proceso de gestión de problemas (*Problem Management*) es responsable de la solución de esta causa raíz [45].

Según ITIL [45], en el proceso de gestión de incidentes, las actividades son las siguientes:

- Identificación del incidente: todos los componentes clave deben ser monitoreados de tal forma que las fallas sean detectadas tempranamente. Esto significa que este proceso puede ser iniciado rápidamente.
- Registro de incidentes: todos los incidentes deben estar registrados sin importar que sean creados a través de una llamada telefónica, detectados automáticamente o que tengan otro origen.
- Categorización del incidente: se necesita encontrar una categorización apropiada para el registro correcto del incidente.
- Priorización de incidente: la priorización puede ser determinada normalmente tomando en cuenta la urgencia del incidente y el nivel de impacto que causa en el negocio. A menudo un indicador del impacto es el número de usuarios afectados.
- Diagnóstico Inicial: Si el incidente ha sido enviado a través de la mesa de servicio, debe tener un diagnóstico inicial, típicamente mientras el usuario está al teléfono y trata de descubrir los síntomas completos del incidente y determinar exactamente qué salió mal y como corregirlo.

Todos los incidentes deben tener un esquema estándar de categorías. Este permite un acceso más rápido a los incidentes e información de solución de problemas, y también permiten un mejor soporte para manejo proactivo de los incidentes. Debido a que las organizaciones son únicas en sus categorías, es difícil encontrar un conjunto universal de las mismas, especialmente en el nivel detallado [46].

ITIL propone los siguientes pasos genéricos para la caracterización de incidentes [45]:

1. Realizar sesiones de lluvia de ideas (*brainstorming*) con los interesados (*stakeholders*).
2. Generar las categorías iniciales incluyendo "otros".
3. Usar las categorías principales por un corto periodo inicial.
4. Analizar los incidentes registrados durante el periodo de prueba. El número de incidentes registrados en cada categoría confirmará si debe existir.

5. Un análisis de incidentes en las categorías dirá si se requieren nuevas categorías subsidiarias.

En operación del servicio de ITIL [45], las categorías de causa raíz responden la pregunta ¿por qué sucedió el incidente?. Para la administración proactiva de problemas, la simple inspección de categorías de incidentes no es una herramienta poderosa. Muchas causas raíz pueden ocasionar síntomas que pueden difundirse a través de varias categorías de incidentes. Por ejemplo, los errores humanos, los problemas de las instalaciones (electricidad, aire acondicionado o humedad), supervisión, guía y problemas en los procesos, causan incidentes en varias categorías. Los incidentes y categorías de causa raíz pueden ser considerados independientes entre sí. Todas las categorías de causa raíz pueden existir en todas las categorías de incidentes.

En la literatura, la clasificación de causa raíz o categorización es poco tratada [46]. No existe una guía en la categorización de causa raíz en ITIL, en contraposición a las categorías de incidentes, por esto se utilizan las de incidentes en el presente trabajo.

Según ITIL [45], en un registro de incidente se pueden incluir cambios, ítems de configuración (*Configuration Item*), otros incidentes relacionados, problemas, cambios o errores conocidos, información de monitoreo y capacidad.

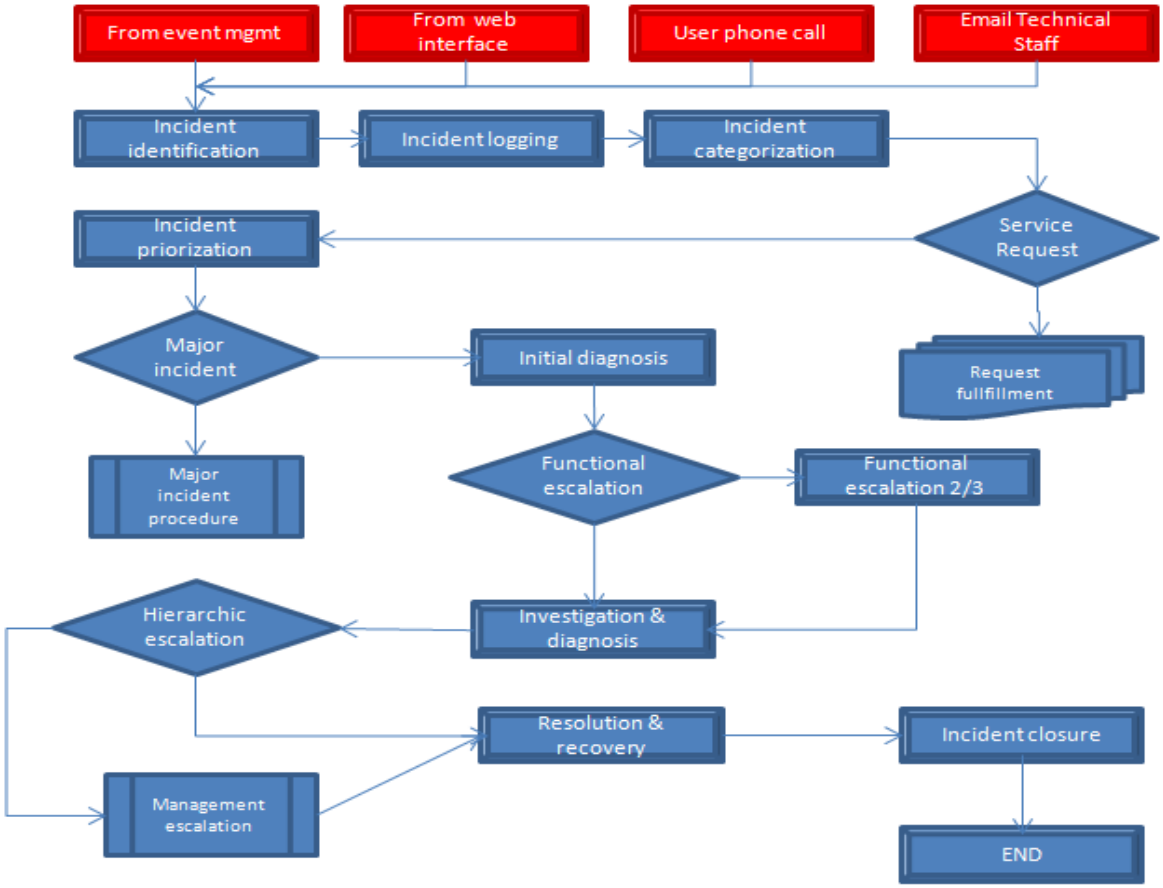

Figura 3. Proceso de gestión de incidentes. Tomado de Tomado de Von Bon, J et al., **[44]**

Según ITIL [45], un incidente se inicia en la mesa de servicio (*service desk*) donde se registra y se le asigna un consecutivo o un *ticket.* El *ticket* se abre, ya sea desde el monitoreo o desde una interfaz de usuario (teléfono, web, email, etc.). Ningún incidente se trabaja sin un *ticket*. El *service desk* es responsable de monitorear el incidente durante su ciclo de vida. Los incidentes que no puedan ser resueltos inmediatamente por el *service desk* (nivel 1) se escalan a los grupos de especialistas apropiados (niveles 2 y 3). Los recursos de nivel 2 y 3 establecen una solución alternativa (*workaround*) o una solución tan pronto como sea posible para restaurar el servicio al cliente. Luego de la solución el incidente se retorna al *service desk*. El *service desk* contacta al usuario para verificar que el servicio haya sido restaurado al nivel acordado. Si es así, se cierra el incidente. Si no

se ha restaurado al nivel acordado, se escala de nuevo a los niveles 2 y 3 con una anotación, de que la solución inicial no fue adecuada.

2.3.2 NOC – *Network Operation center*

El centro de operaciones de red (de la sigla en inglés *Network Operation Center*) es un lugar donde los técnicos de TI pueden directamente soportar los esfuerzos de monitoreo remoto y gestión [47]. Actúa como el punto de contacto para todas las peticiones relacionadas con redes, mantenimiento y solución de problemas.

2.3.3 Sistemas de gestión de incidentes

Según Latrache [48], construir un sistema de gestión de incidentes de acuerdo a los lineamientos ITIL no es una tarea trivial. Existen soluciones como BMC Footprints [49] que provee un conjunto de herramientas para empresas medianas, Remedy [50], una plataforma configurable y extensible para llevar seguimiento de incidentes, y NeTSupport [51], la plataforma online más difundida para gestionar incidentes y problemas de acuerdo a ITIL. Estas soluciones son programas para mesas de ayuda basados en web y no cubren eficientemente el ciclo de vida de un incidente [48].

Para mejorar la calidad del proceso de gestión de incidentes, se han propuesto varios enfoques no comerciales. Estos enfoques utilizan técnicas como minería de datos y algoritmos de toma de decisiones [52]. Así mismo, los sistemas multi-agente surgen como una alternativa para implementar procesos de gestión de incidentes gracias a su eficiencia, robustez y confiabilidad [48].

Debido a que el presente trabajo utiliza un sistema multi-agente para diagnosticar los problemas de un servicio, se incluyen a continuación los conceptos fundamentales de estos sistemas.

2.4 Agentes inteligentes

2.4.1 Agentes

Según Wooldridge [16], un agente es un sistema computacional que está situado en algún ambiente y que es capaz de acción autónoma en este ambiente para alcanzar sus objetivos de diseño. El ambiente es todo lo externo al agente. El ambiente puede ser físico (por ejemplo, un sistema de control) o puede ser un ambiente computacional (por ejemplo, fuentes de datos, recursos de cómputo u otros agentes). Un agente puede alterar el ambiente tomando alguna acción. La separación de los agentes del ambiente significa que los agentes son distribuibles. Bajo la definición de Wooldridge, una entidad situada en un ambiente es un agente si puede actuar autónomamente en respuesta a los cambios ambientales. La definición de autonomía indica que un agente "Ejercita control de sus propias acciones" [53], indicando esto que puede iniciar o programar ciertas acciones para ser ejecutadas.

Wooldridge y Jennings [54] identificaron tres tipos de agentes:

- Agentes que ejecutan tareas directas basadas en reglas y supuestos predefinidos.
- Agentes que ejecutan una tarea bien definida a petición del usuario.
- Agentes que entregan información o servicios a un usuario cuando sea apropiado, sin que explícitamente se les pida hacerlo.

Según Wooldridge [55], los agentes Inteligentes son aquellos que muestran una autonomía flexible, es decir, que posean las características de reactividad, proactividad y habilidad social, que se describen a continuación:

- Reactividad: un agente inteligente reacciona a los cambios en su ambiente oportunamente.
- Proactividad: los agentes inteligentes tienen un comportamiento orientado a metas. Este comportamiento supone que un agente cambiará su comportamiento dinámicamente para alcanzar sus metas. Wooldridge [16] describe esto como la habilidad de "tomar la iniciativa".
- Habilidad social: los agentes inteligentes son capaces de interactuar con otros agentes inteligentes. La habilidad social incluye mucho más que intercambiar

mensajes entre entidades diferentes de hardware y software. Denota la habilidad de negociar e interactuar de una forma cooperativa. Esta habilidad se crea normalmente por medio de un lenguaje de comunicación de agentes (ACL, *agent communication language*), el cual les permite a los agentes conversar en lugar de simplemente intercambiar datos.

2.4.2 Sistemas multi-agente

Un sistema multi-agente es un sistema que comprende dos o más agentes inteligentes [54]. Aunque no existe una meta global del sistema, cada agente por separado tiene su propia meta local [54]. Dependiendo de la definición de agente, los agentes en un sistema multi-agente pueden o no tener la habilidad de comunicarse entre sí directamente. Sin embargo, bajo las definiciones de Wooldridge [16], los agentes inteligentes deben tener la habilidad social y, por lo tanto, ser capaces de comunicarse entre sí.

Los MAS han sido explotados de dos formas, como un enfoque para construir sistemas de hardware/software flexibles y extensibles y como un enfoque de modelado [54]. También se han utilizado para construir sistemas robustos, flexibles y extensibles, apoyados en las características de autonomía, arquitecturas de MAS abiertas, plataformas para sistemas distribuidos y tolerancia a fallos.

Según Wooldridge [55], un agente encapsula un conjunto de funcionalidades, de manera similar a la programación orientada a objetos. Pero el agente es autónomo en su decisión de atender o no la petición que le hayan hecho y con qué prioridad. Esto es muy útil cuando el agente está recibiendo muchas peticiones y no puede responderlas en un tiempo razonable. Como los agentes no están directamente unidos a otros, se puede sacar uno de la operación o adicionar otro mientras los demás siguen ejecutándose, esto permite la flexibilidad y la extensibilidad. Un marco de trabajo (*framework*) de agentes provee la funcionalidad para el envío de mensajes y la localización de los servicios; esto significa que la integración y comunicación de un nuevo agente son manejadas sin esfuerzo por parte del diseñador del sistema.

2.4.3 Clasificación de arquitecturas de sistemas multi-agente

Según Weyns, D. [56], la base del mecanismo de razonamiento de un agente se encuentra en la arquitectura del mismo, es el molde para construir un agente, así como la clase lo es a un objeto en la programación orientada a objetos. La arquitectura de agente es el cerebro de un agente porque determina como el conocimiento/información es representado en el agente. También determina la acción que debe tomar el agente basado en su mecanismo de razonamiento/interpretación. Así, diferentes arquitecturas usan diferentes enfoques de representación para que su mecanismo de razonamiento pueda resolver problemas variados.

Estas arquitecturas pueden ser categorizadas en líneas generales en tres grupos: arquitectura clásica, cognitiva y semántica [57]. La arquitectura clásica incluye la basada en lógica, la reactiva, BDI (*Belief Desire Intention*) y la híbrida. La basada en lógica utiliza representación simbólica para el razonamiento. La reactiva utiliza representación como estímulo-respuesta. De otro lado, la BDI está basada en estados mentales característicos como creencias, deseos e intenciones. La arquitectura en capas es la híbrida de reactiva y deliberativa. La arquitectura cognitiva está basada en ciencias cognitivas y la arquitectura semántica utiliza tecnología semántica.

2.4.4 Arquitecturas basadas en lógica

Según Kim O. et al. [57], la arquitectura basada en lógica, también conocida como basada en símbolos o deliberativa, es una de las más tempranas arquitecturas de agentes que se basa en el enfoque tradicional simbólico artificial que representa y modela el ambiente y el comportamiento del agente con representaciones simbólicas.

El rol de un agente puede ser considerado como de probador de teoremas. La manipulación sintáctica de la representación simbólica es el proceso de deducción lógica o prueba de teoremas. Como una instancia de la prueba de teoremas, las especificaciones de un agente indican cómo se comporta, como se generan los objetivos y qué acción debe tomar para alcanzar estos objetivos [57].

A pesar de la simplicidad y elegancia de la semántica lógica de la arquitectura es atractiva, existen problemas asociados a este enfoque. Primero, el problema de transducción implica que el problema de traducir el modelo a representación simbólica. Segundo, es difícil representar información de una forma simbólica con la que los agentes puedan razonar y en un ambiente restringido en el tiempo. Finalmente, la transformación de las entradas percibidas, tal vez no sea lo suficientemente precisa para describir el ambiente debido a fallas como errores de los sensores, errores de razonamiento y otros [57].

Construir un agente en este enfoque se ve como un proceso deductivo. Se codifica un agente como una teoría lógica usando especificación y el proceso de seleccionar una acción es a través de un proceso deductivo que reduce el problema a una solución tal como en prueba de teoremas [57].

2.4.5 Arquitectura reactiva

Según Kim O. et al. [57], la arquitectura reactiva, se basa en la relación de una situación con una acción. Es diferente de la basada en lógica porque no existe un modelo del mundo ni se una razonamiento simbólico. Un agente responde a los cambios en el ambiente siguiendo un esquema estímulo-respuesta. La arquitectura es lograda a través de un conjunto de sensores y actuadores, donde las entradas percibidas se relacionan con actuadores y con cambios en el ambiente.

Una de las ventajas es que es menos complicada de diseñar e implementar que la arquitectura basada en lógica. El comportamiento del agente se puede rastrear computacionalmente. Otra ventaja es la robustez ante fallos. Por medio de la interacción de comportamientos simples se pueden alcanzar los complejos [57].

2.4.6 Arquitectura BDI (*Belief-Desire-Intention*)

Según Kim O. et al. [57], se basa en el razonamiento práctico con énfasis en la intención de Bratman. El razonamiento práctico es razonamiento dirigido a acciones, el proceso de dilucidar qué hacer. Esto difiere del proceso de razonamiento teórico porque deriva

conocimiento o llega a conclusiones usando las creencias y el conocimiento propios. El razonamiento práctico humano incluye dos actividades, la deliberación y el razonamiento ruta-objetivo. La deliberación decide cuál estado se requiere alcanzar, mientras que el razonamiento ruta-objetivo decide cómo alcanzar este estado.

En la arquitectura BDI, el agente consiste de tres componentes lógicos que representan los estados mentales, a saber creencias, deseos e intenciones. Las creencias son el conjunto de información que un agente tiene acerca del mundo. Los deseos, son la motivación o posibles opciones para llevar a cabo las acciones. Las intenciones son los compromisos del agente hacia sus deseos y creencias. Las intenciones son componentes clave en el razonamiento práctico. Ellas describen los estados que el agente se ha comprometido a alcanzar y como resultado, inducen a la acción. La formación de estas intenciones es crítico para el éxito del agente [57].

Las ventajas de esta arquitectura es que el diseño es claro e intuitivo. La descomposición funcional del subsistema de agente es clara y la lógica BDI tiene propiedades lógicas que pueden ser estudiadas. Sin embargo, en la arquitectura BDI, no es claro cómo implementar eficientemente la funcionalidad del subsistema y, por esto, los agentes deben alcanzar un balance entre su compromiso y la reconsideración. Si un agente no se detiene a reconsiderar, puede que esté intentando lograr una intención que no es lograble o que ya no es válida. Si un agente reconsidera frecuentemente, puede enfrentar el riesgo de no alcanzar sus intenciones debido que no tiene suficiente tiempo para trabajar en la tarea [57].

2.4.7 Arquitectura en capas (híbrida)

Según Kim O. et al. [57], esta arquitectura permite comportamientos tanto reactivos como deliberativos. Combina las ventajas de las arquitecturas reactiva y la basada en lógica y al mismo tiempo alivia los problemas de ambas arquitecturas. Los subsistemas son descompuestos en capas de estructura jerárquica para tratar con diferentes comportamientos. Existen dos tipos de interacciones que fluyen entre las capas, el horizontal y el vertical. En la horizontal, cada capa está directamente conectada a las

entradas de los sensores y la salida de los actuadores. Cada capa es como un agente que relaciona la entrada a la acción a ser tomada.

La arquitectura de capa vertical elimina algunas de las dificultades de la horizontal, debido a que la entrada de los sensores y la salida a los actuadores son cada uno tratados por una capa individual. Existen dos tipos de arquitecturas de capas verticales, la de control de una-pasada y de dos-pasadas. La de una-pasada el control fluye desde la capa inicial hacia los sensores y hasta la capa final que genera la acción. En la arquitectura de dos-pasadas, los datos fluyen arriba en la secuencia de capas y el control entonces fluye hacia abajo [57].

2.4.8 Arquitectura cognitiva

Según Kim O. et al. [57], se basa en la parte de la cognición, la simulación y el modelado del comportamiento humano. Estas arquitecturas se usan para construir sistemas inteligentes que modelen el desempeño humano. Esta arquitectura cognitiva debe poseer las características:

- memorias de corto y largo término para almacenar las creencias, objetivos y conocimiento del agente.
- representación y organización de las memorias
- procesos funcionales que operan sobre estas estructuras

Esta arquitectura se diferencia del enfoque multi-agente en diferentes maneras, [57]:

- la arquitectura cognitiva viene con un formalismo para codificar el conocimiento y asociarlo con su intérprete.
- tiene fuertes suposiciones acerca de la representación del conocimiento y el proceso que opera sobre el mismo.
- asume una representación modular del conocimiento,
- ofrece un comportamiento inteligente al nivel del sistema, en lugar de al nivel de métodos componentes diseñados para tareas específicas y
- provee un enfoque unificado en el cual un conjunto de representaciones comunes y mecanismos reducen la necesidad de crearlas detalladamente.

2.4.9 Estructura de un sistema multi-agente

Según Millet, S. et al. [58], la arquitectura de un sistema multi-agente puede ser vista como un caso especial de la arquitectura contenedor-componente. En este caso, los componentes son agentes y el contenedor es un ambiente que provee servicios de comunicación y descubrimiento a sus agentes.

Un agente es un componente activo y orientado a objetivos (*goal-oriented*) que juega uno o más roles en el ambiente [58]. Un agente se encuentra en un ambiente determinado. El ambiente también se denomina contenedor. Los roles pueden ser simples o compuestos. El rol compuesto tiene asociado una máquina de estado finito. El rol simple puede ser cíclico o de una sola ejecución. Estas relaciones se pueden observar en la figura 4.

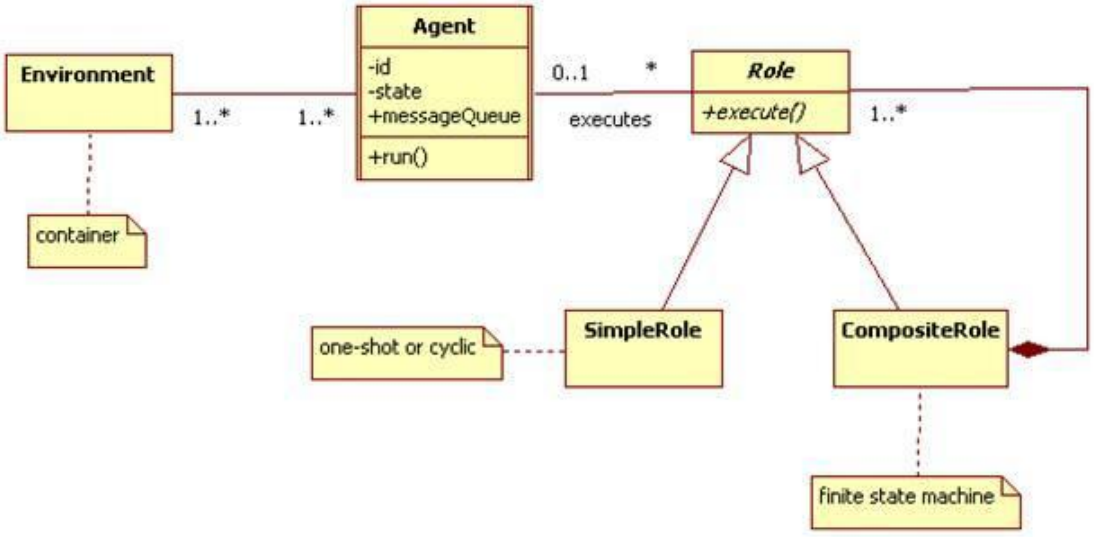

Figura 4. Estructura de un sistema multi-agente. Tomada de Millet, S. et al., **[58]**

Según Millet, S. et al. [58], cada rol ejecuta un ciclo de control como el siguiente:

1. observar el estado del ambiente
2. si el estado del ambiente es el estado objetivo, finalice
3. seleccione una acción
4. realice una acción
5. retorne a 1.

Un agente puede observar todo o parte del ambiente en el paso 1. Esto puede implicar observar los estados de otros agentes en el ambiente. Cada agente puede tener diferentes objetivos. La acción seleccionada en el paso 3 depende de la observación hecha en el paso 1. También puede depender del estado interno del agente. Este estado interno puede ser una memoria de un bit, o una compleja base de conocimiento. Realizar la acción en el paso 4 modifica el estado del ambiente. También puede modificar el estado interno del agente y sus objetivos. La acción puede incluir el envío de mensajes a otros agentes. Nuevos roles pueden ser adicionados a la lista de roles. Adicionalmente, los roles pueden extraer mensajes de cola de mensajes y pueden adicionar nuevos mensajes a las colas de mensajes de otros agentes, [58].

2.4.10 Arquitectura estándar de la FIPA

La arquitectura de referencia estándar para MAS es la de FIPA [18] (*Foundation for Intelligent Physical Agents*). Según Poslad, S. [59], FIPA es una organización de estándares de la *Computer Society* de *IEEE*, que promueve la tecnología basada en agentes y la interoperabilidad de sus estándares con otras tecnologías. El estándar FIPA cuenta con diferentes implementaciones como JADE y Zeus.

Según Poslad, S. [59], las arquitecturas abiertas de los MAS no ponen restricciones en el lenguaje de programación del sistema y permiten comunicación flexible entre todos los agentes. El modelo de referencia de gestión de agentes incluye el marco de trabajo y define los estándares para crear, localizar, remover y comunicarse de los agentes. Esta es llamada generalmente la plataforma de agentes y es una parte del ambiente de los agentes.

Según Poslad, S. [59], para la FIPA, un MAS la estructura incluye un AMS, DF, y un MTS. El AMS (*Agent Management System)*, un agente que provee servicios de páginas blancas además de servicios de ciclo de vida y administración de recursos, el DF *(Directory Facilitator)*, es un agente que provee servicios de páginas amarillas, MTS (*Message Transport System*) también llamado canal de comunicación de agentes, provee servicios

de comunicación para intercambio de mensajes entre agentes. Dicha estructura se puede apreciar en la figura 5.

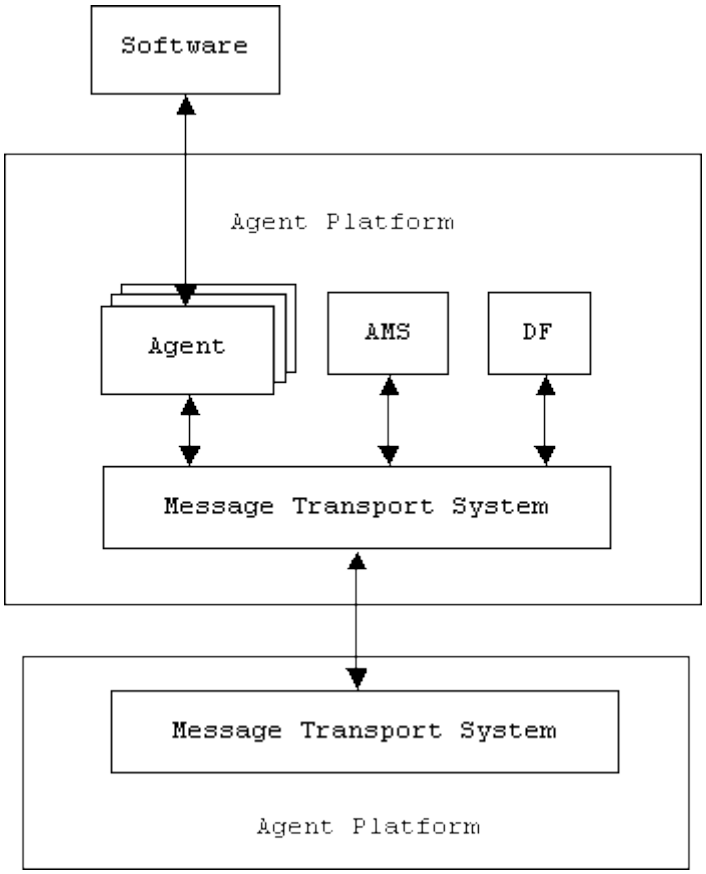

Figura 5. Estructura estándar para sistemas multi-agente FIPA. Tomada de Poslad, S. **[59]**.

Según Poslad, S. [59], en FIPA un agente es un proceso computacional que implementa uno o más roles en una aplicación. Un agente puede ser implementado como un Java Bean, un objeto COM u otras entidades de software. Los agentes se comunican usando un ACL (*Agent Communication Language*), Un agente tiene nombre, atributos y una forma de ser localizado. Los agentes se pueden registrar o des-registrar en varios servicios de directorio o páginas amarillas. Los agentes pueden buscar a otros en este servicio de directorio.

Las interacciones de agentes en FIPA se denominan conversaciones. Una conversación es un intercambio de mensajes ACL, [59]. Un mensaje ACL puede ser de uno de los tipos presentados en la tabla 1.

Tipo de Mensaje	Nombre en FIPA
Solicitud de propuestas	*Call for proposal*
Aceptar la propuesta	*Accept proposal*
Aceptación	*Agree*
Cancelar	*Cancel*
Confirmar	*Confirm*
NO Confirmar	*Disconfirm*
Falla	*Failure*
Informar	*Inform*
Informar condicional	*Inform-If*

Tabla 1. Tipos de mensajes en FIPA. Tomada de Poslad, S. **[59]**.

FIPA también define varios protocolos de interacción tal como el *Contract Net Interaction Protocol*. Además, es posible que los agentes acuerden en el uso de un vocabulario común para facilitar las conversaciones, esto se conoce como la ontología del sistema [59].

La plataforma no debe poner restricciones a la creación y los mensajes de los agentes, además debe tener un mecanismo para localizar agentes que ofrezcan servicios específicos. En el modelo FIPA esto se logra por medio de un agente denominado el *Directory Facilitator* [48].

Según Wooldridge, M. [16], los agentes se pueden organizar en plataformas para agentes distribuidos. Debido a que conservan sus metas y habilidades a pesar de estar en diferentes ambientes. Esto los hace capaces de ser distribuidos en redes por medio de la plataforma de agentes. Esta puede instalarse en varios computadores e incluso en dispositivos móviles. En una plataforma no se diferencia entre agentes en el mismo dispositivo o en otros dispositivos. Esto significa que el mismo conjunto de agentes puede ser instalado en uno o en múltiples dispositivos conectados en red sin necesidad de modificar o cambiar el código del agente.

Para hacer un sistema tolerante a fallos se busca dotar de redundancia a los sistemas. Para lograr esto en los MAS se debe proveer más de un agente con el mismo conjunto de habilidades. En caso de un fallo en un agente, el *Directory Facilitator* permite encontrar un nuevo agente capaz de responder a la petición. Entonces, la tolerancia a fallas se logra a través de la duplicación de agentes y su distribución a través de diferentes dispositivos [54].

Según On, K. et al. [57], un MAS se caracteriza por la estructura interna de los agentes y la que existe entre ellos. Al nivel de agentes individuales, se han desarrollado muchas arquitecturas diferentes, que van desde agentes solo reactivos, hasta complejos agentes que realicen razonamientos. En el nivel de sistema, el MAS puede ser estructurado como una organización de agentes egoístas que juegan diferentes roles en la organización, persiguiendo sus propios intereses. Otros MAS consisten de agentes cooperativos que buscan alcanzar un fin común. Los agentes pueden interactuar en formas diferentes: por medio de un lenguaje de alto nivel y protocolos de interacción o por medio de marcas en un medio de coordinación.

En el presente capítulo se presentaron los conceptos básicos de gestión de incidentes en ITIL así como de los sistemas multi-agente. En el próximo capítulo se presenta un análisis de los incidentes más frecuentes en servicios de internet, en una empresa de telecomunicaciones.

3. Análisis de incidentes más frecuentes en servicios de acceso a internet corporativo

3.1 Motivación

En este capítulo se determinan los diagnósticos a ser realizados por el sistema. Para ello, se parte de una revisión de los incidentes y los diagnósticos realizados por los expertos humanos (ingenieros de soporte), se agrupan por categorías, se identifican los diagnósticos que tengan asociados el mayor número de incidentes y se estos, se seleccionan los diagnósticos que a criterio de los expertos humanos sean obtenibles por medio de la ejecución de comandos en los dispositivos asociados al servicio.

Para iniciar el proceso, se debe contar con datos de incidentes provenientes de la operación de empresas de telecomunicaciones. De estos datos se obtendrán las frecuencias asociadas a la ocurrencia de cada uno de los diagnósticos. La determinación de incidentes con la mayor frecuencia, de las causas raíz y la organización jerárquica de las categorías permite tener un sistema que cubra la mayoría de los incidentes y que puede constituirse en una herramienta útil a las empresas que provean servicios de internet corporativo, esto se realiza siguiendo las mejores prácticas del proceso de gestión de incidentes ITIL [45].

Debido a que algunas de las gráficas proceden de la empresa donde se realizó el trabajo, no se tiene control de la ortografía que utilizan en sus títulos y denominaciones, por lo que se pide hacer caso omiso de cualquier error ortográfico que se encuentre.

3.2 Obtención de datos

Las empresas modernas soportan sus procesos de gestión de incidentes en aplicaciones informáticas [42]. La información registrada en la atención de un incidente depende de las

características de cada aplicación. La mayoría de ellas permite clasificar los incidentes jerárquicamente. Esta información almacenada se convierte en una fuente de estadísticas y posibles entradas para procesos como el de gestión de problemas.

En el presente trabajo, se toma información de cinco meses de los incidentes en servicios de internet registrados en el sistema de gestión de relaciones con los clientes del Sistema gestión de relaciones con el cliente, CRM por su sigla en inglés, de una empresa de telecomunicaciones. Esta información debe tener un manejo confidencial.

La clasificación de los incidentes y su diagnóstico varía durante su ciclo de vida, [45]. Por ello, la información suministrada permite conocer el diagnóstico definitivo asociado al incidente y establecer relaciones entre clasificaciones de los incidentes y sus diagnósticos.

A criterio de los expertos humanos, las columnas relevantes para la labor de clasificación de los incidentes son: consecutivo, tipo de servicio, localización, elemento, causa-raíz, fecha-registro y tiempo-de-atención. Un fragmento de esta información se puede observar en la figura 6.

P	Q	R	S	T
CIUDAD INCIDENTE DESTINO	Tipo	Localización	Elemento	Causa raiz
Bogotá	Corporativo	Equipo Remoto	Router cliente	Dañado
Bogotá	Corporativo	Equipo Remoto	Teléfono IP	Dañado
Bucaramanga	Corporativo	Equipo Remoto	Modem cliente	Dañado
Cali	Corporativo	Equipo Remoto	Modem cliente	Dañado
Bucaramanga	Corporativo	Equipo Remoto	Router cliente	Dañado
Medellín	Corporativo	Equipo Remoto	Modem cliente	Dañado
Funza	Corporativo	Equipo Remoto	Modem cliente	Dañado
Medellín	Corporativo	Equipo Remoto	Modem cliente	Dañado
Bucaramanga	Corporativo	Equipo Remoto	Modem cliente	Dañado
Bogotá	Corporativo	Equipo Remoto	Modem cliente	Dañado
Bogotá	Corporativo	Equipo Remoto	Modem cliente	Dañado
Buenaventura	Corporativo	Equipo Remoto	Modem cliente	Dañado
Cali	Corporativo	Equipo Remoto	Teléfono IP	Dañado

Figura 6. Fragmento de los datos analizados. Tomado de la empresa de telecomunicaciones

3.3 Análisis descriptivo de los datos

Según Shumway, R. H. et al. [60], la correlación introducida por el muestreo de puntos adyacentes en el tiempo, puede restringir la aplicabilidad de muchos métodos estadísticos que dependen del supuesto que estas observaciones adyacentes son independientes e idénticamente distribuidas. Por ello, en el presente trabajo, se realiza un análisis descriptivo de los datos así como un análisis de series de tiempo, siguiendo el procedimiento descrito en Kerns, J. et al. [61]. Para la cantidad de incidentes diarios, se pretende determinar si existen comportamientos cíclicos, irregulares o tendencias, ya sean de tipo creciente o decreciente. Para el análisis de los datos se utilizó el lenguaje R [62]. R es un lenguaje y un ambiente para cálculo de estadísticas y generación de gráficas.

3.3.1 Cantidad de incidentes diarios

Se construye una serie de tiempo con la cantidad diaria de incidentes registrados durante cinco meses en un NOC de una empresa de telecomunicaciones, los detalles de la muestra se presentan en la figura 7. La serie de tiempo se presenta en la figura 8.

Característica	Valor
Ámbito:	Servicios corporativos
Universo:	Incidentes en servicios
Tamaño:	15.527
Periodo:	2014-06-01 a 2014-10-31

Figura 7. Ficha de la muestra tomada. Creada por el autor.

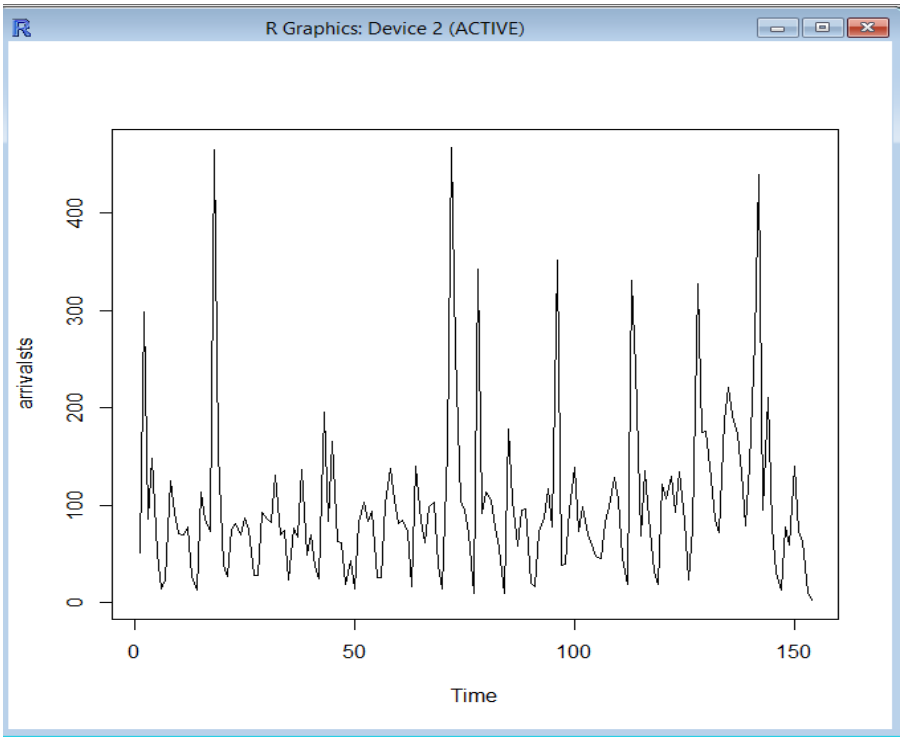

Figura 8. Serie de tiempo de llegadas de incidentes. Creada por el autor usando R a partir de los datos de la empresa de telecomunicaciones.

El enfoque de dominio de tiempo es generalmente motivado por la presunción que la correlación entre puntos adyacentes en el tiempo se explica mejor en términos de una dependencia del valor actual en valores pasados. Existen dos enfoques uno que supone que los datos observados son resultado del producto de factores que envuelven operadores diferenciales o ecuaciones diferenciales respondiendo a una entrada de ruido blanco, mientras que el otro asume que es el resultado de la suma de series, cada una con una estructura especificada de series de tiempo, por ejemplo, en economía, se asume que una serie se genera como la suma de series tendencia, estacional, aleatoria y error [60].

En el presente trabajo, se sigue el enfoque de suma de series, para la serie de tiempo de llegada de incidentes, aquí, la serie de tiempo se descompone en tres tipos de patrones de series de tiempo: tendencia, periódico y aleatorio. Tendencia (*trend)*, cuando existe un incremento o decremento en una semana o más, el patrón periódico (*seasonal),* cuando una serie se influencia por el día de la semana u otros factores temporales, el patrón aleatorio (*random)* irregular o ruido, describe los residuos de las series de tiempo luego de que los otros componentes han sido removidos. El detalle de esta descomposición se observa en la figura 9.

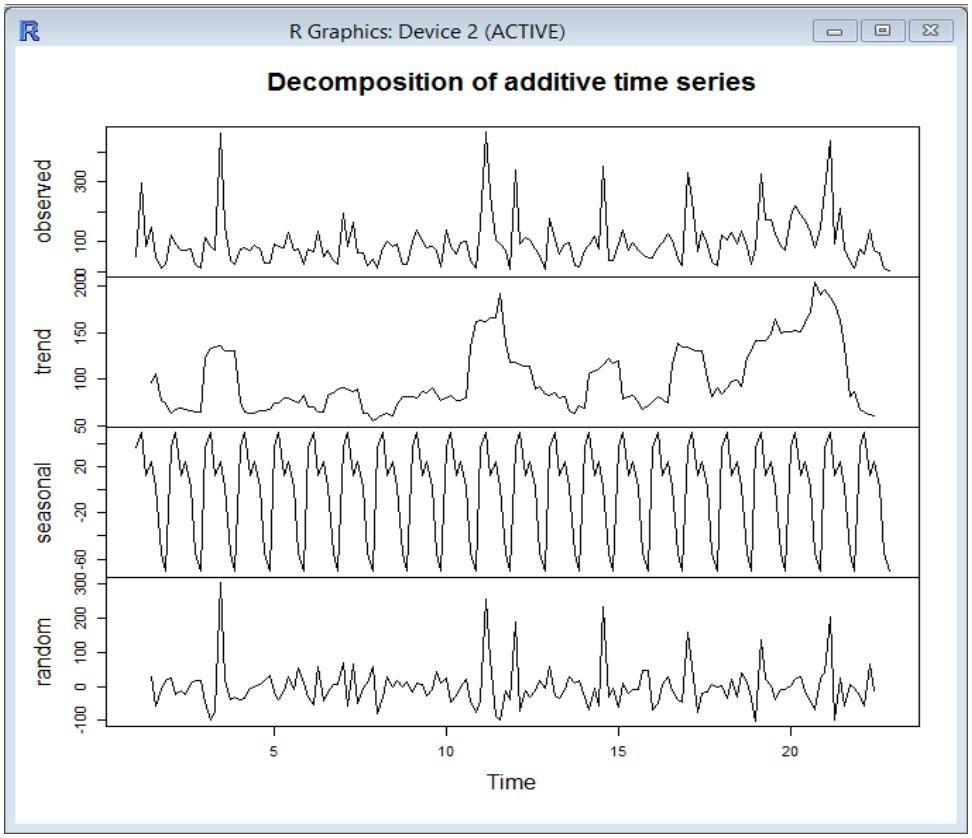

Figura 9. Descomposición de la serie de tiempo de llegadas de incidentes. Creada por el autor en R. a partir de los datos de la empresa de telecomunicaciones

El patrón *trend* se puede asociar a problemas debidos a cambios en la red, migraciones de equipos, que puedan aumentar o disminuir los incidentes.

El patrón *seasonal* se asocia a los días de la semana, siendo el domingo el día con menor número de incidentes registrados.

El patrón *random* se asocia a fallas masivas de la red, ya sea por cortes de fibra o fallas en dispositivos. Según los expertos humanos, los picos observados del 2014-09-09 se debieron a daños en un puerto de un *router* PE, 2014-07-17 bloqueo de tarjeta en un *switch*, 2014-11-18 daños en puerto en un *router* PE, 2014-10-03 error humano de un proveedor, 2014-09-15 daño *transceiver* en proveedor, 2014-10-20 ruptura de fibra.

3.3.2 Tiempo de solución de incidentes

Los tiempos de solución de incidentes se presentan en la figura 10. Se observa como la mayoría de incidentes son solucionados en la primera hora de su registro.

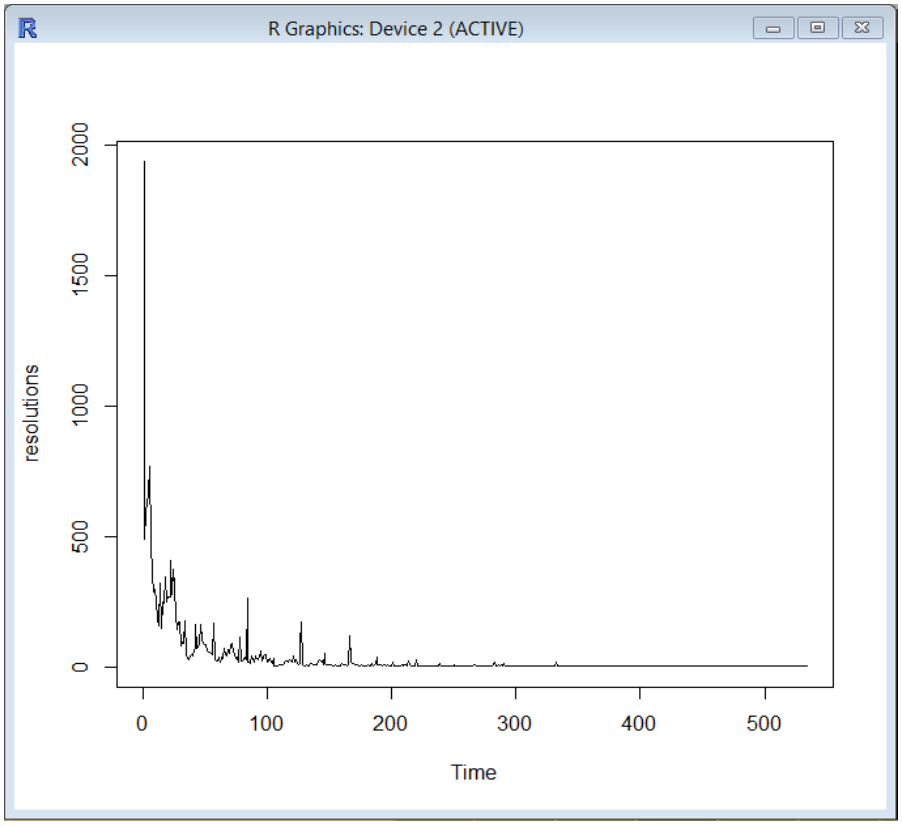

Figura 10. Distribución de los tiempos de atención de incidentes. Creada por el autor en R. a partir de los datos de la empresa de telecomunicaciones

En resumen, el tiempo de atención promedio es de 33.31 horas, la desviación estándar es de 118.60 y el 75% de los incidentes son atendidos en 13.75 horas o menos.

3.3.3 Categorías de clasificación de incidentes

Las categorías de clasificación suministradas por el personal de soporte de la empresa de telecomunicaciones, presentan una organización jerárquica: en un primer nivel, tipo de servicio (corporativo, pyme); en el segundo nivel, localización (PE, el *router* de borde del proveedor, *switch* de acceso, CPE equipo del lado del cliente, procedimiento y otros); en el tercer nivel, elemento de red con marca (cisco, Huawei y otras); y en el cuarto nivel, causa raíz. A partir de los datos de los incidentes se realiza una representación gráfica de las relaciones jerárquicas entre las categorías. Esto se puede observar en la Figura 11.

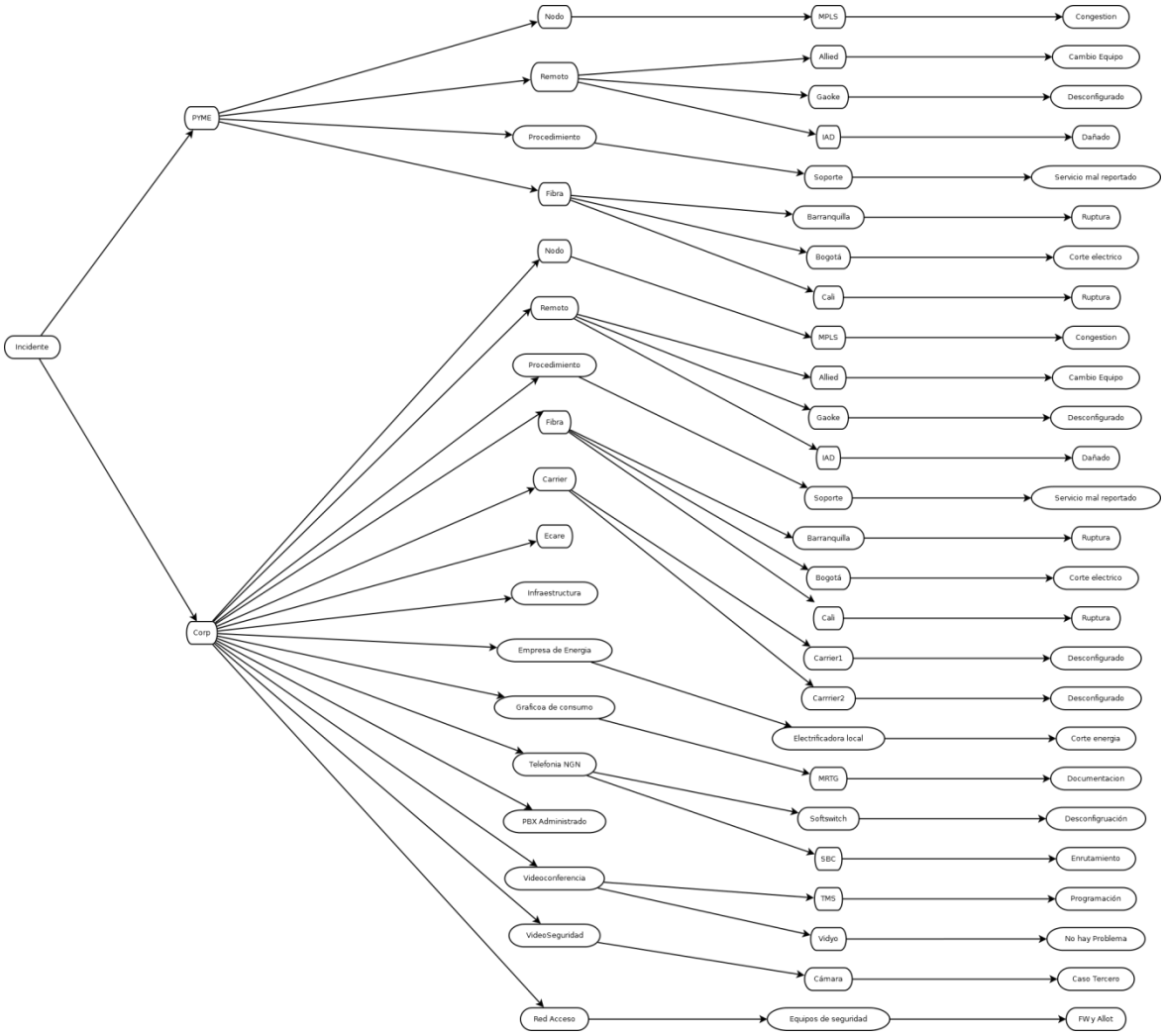

Figura 11. Clasificación de incidentes encontrada. Creada por el autor a partir de los datos de la empresa de telecomunicaciones

Usando la localización del incidente, como criterio de agrupación, los tipos de incidentes más numerosos son los de "Fibra" y "Equipos remotos". La figura 12 muestra las cantidades de cada tipo de incidente.

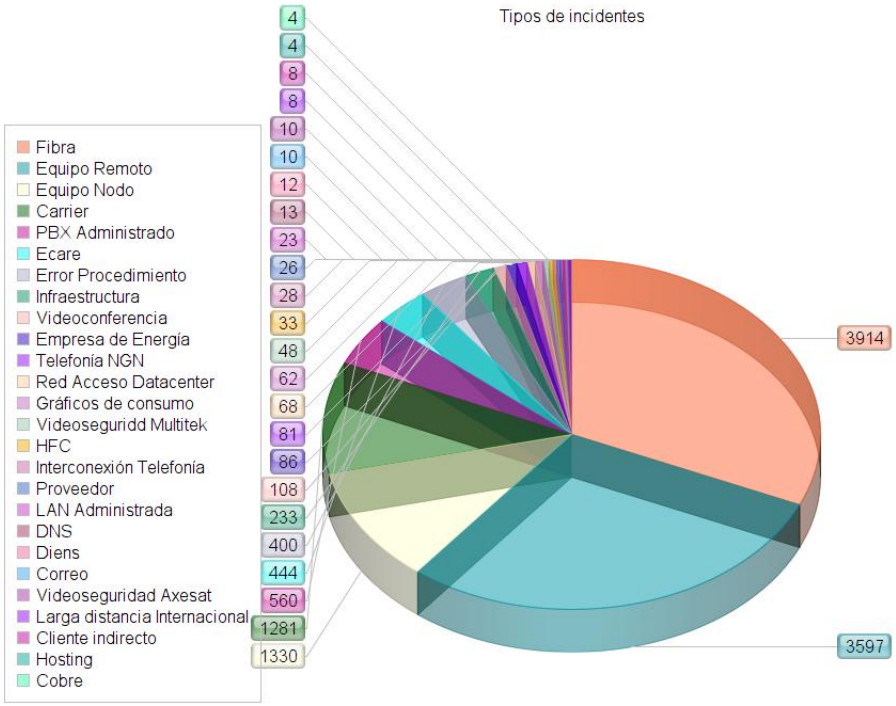

Figura 12. Distribución de tipos de incidentes. Los números indican la cantidad de incidentes por cada tipo. Tomado de la empresa de telecomunicaciones.

3.4 Diagnósticos a realizar

3.4.1 Selección de diagnósticos

Según el criterio de los expertos humanos, (los ingenieros NOC entrevistados) los diagnósticos a considerar son las causas raíz asociadas a la mayor cantidad de incidentes, y, de éstos, los que puedan ser obtenibles por medio de la ejecución de comandos en los dispositivos de la red.

Del total de incidentes agrupados por causa raíz, organizado por cantidad de mayor a menor, se seleccionan los primeros 16 que incluyen el 80,29% de los incidentes (12467). Estas causas raíz y el número de incidentes para cada tipo se pueden observar en la figura 13.

Causa raíz	Frecuencia	Frec.Acum.	Porcentaje	Porcentaje Acum.
Falla fibra	3118	3118	20,08114897	20,08115
Bloqueo CPE	2048	5166	13,18992722	33,27108
Error configuración puerto	1555	6721	10,01481291	43,28589
CPE dañado	1038	7759	6,68512913	49,97102
Interfaz caida	624	8383	4,018805951	53,98982
Corte eléctrico	581	8964	3,741869002	57,73169
Daño por vandalismo	510	9474	3,284601018	61,01629
Falla nodo	458	9932	2,949700522	63,96599
Falla switch	423	10355	2,724286726	66,69028
CPE averiado	415	10770	2,672763573	69,36305
Atenuación	370	11140	2,382945836	71,74599
Cuentas	337	11477	2,170412829	73,9164
Falla eléctrica	306	11783	1,970760611	75,88716
Tarjeta bloqueada	264	12047	1,700264056	77,58743
Condiciones climáticas	219	12266	1,410446319	78,99787
Tabla ARP vacía	201	12467	1,294519225	80,29239
Extensión	168	12635	1,081986218	81,37438
Actualización firmware	153	12788	0,985380305	82,35976
Apagado	132	12920	0,850132028	83,20989
Intermitencia	123	13043	0,792168481	84,00206
Desconectado	113	13156	0,727764539	84,72983
Enrutamiento	112	13268	0,721324145	85,45115

Figura 13. Causas raíz con cantidad de incidentes y porcentajes del total. Tomado de la empresa de telecomunicaciones.

Según los expertos humanos, del total de causas raíz, se deben tomar aquellas que se puedan obtener por medio de la ejecución de comandos en los elementos de red, éstas se presentan en la figura 14.

Diagnóstico
Ultima milla OK
Bloqueo
Error Configuración
Falla Fibra
Falla Nodo
Tabla ARP vacia
Interfaz caida
Error Configuración Puerto
Falla Switch

Figura 14. Lista de diagnósticos. Tomado de la empresa de telecomunicaciones.

3.4.2 Proceso actual de diagnóstico

Para conocer el proceso actual de diagnóstico se realizaron entrevistas a los expertos humanos, los ingenieros NOC (ingenieros encargados de solucionar los incidentes). Adicionalmente, se revisó la documentación escrita representada en los manuales de atención de incidentes. Como resultado de esta tarea, se obtuvo un listado de comandos a ejecutar en los dispositivos de red y un listado de reglas a aplicar.

Un servicio de acceso a internet corporativo requiere la configuración en por lo menos tres elementos: un *router* de borde o PE (*Provider Edge*), un *switch* de acceso y un *router* de cliente o CPE (*Customer Premises Equipment*).

Para un servicio de internet, a nivel del PE, se configura una interfaz y una ruta que permitirá la salida de los dispositivos del cliente hacia internet.

En el PE se ejecutan comandos para conocer la respuesta de la dirección IP WAN (Wide Area Network) del CPE, la herramienta utilizada es *ping*. Esta prueba se conoce como "última milla" y se identifica con el código WAN_OK.

Si el equipo tiene una dirección IP de gestión también se ejecuta un comando para conocer la respuesta por medio de la herramienta *ping* hacia esta IP. Esta prueba se conoce con el código GEST_OK.

Para determinar si el PE se encuentra funcionando se ejecuta un comando para conocer su respuesta usando la herramienta *ping* hacia su dirección IP. Esta prueba se conoce con el código NODE_DOWN.

A nivel del *switch* de acceso, se configura una interfaz, una VLAN (*Virtual LAN*) y un modo de operación. Aquí se ejecutan comandos de verificación de las direcciones físicas o MAC (*Media Access Control*) registradas por el puerto. Esta prueba se conoce con el código REGISTERING_MAC. Cuando la conexión de última milla utiliza fibra óptica, se requiere configurar dispositivos transmisores/receptores o TRV (*Transceivers*). Si la tecnología de acceso es una PON (Passive Optical Network), se configuran OLT (Optical Line Terminal) del lado del proveedor y ONU(Optical Network Unit) cerca de los clientes.

Para obtener el estado de la interfaz en el *switch* se ejecuta un comando identificado con el código PORT_REGISTERED.

Para determinar si el *switch* de acceso se encuentra operativo, se ejecuta un comando identificado con el código SWT_DOWN.

A nivel del CPE configura un direccionamiento IP, una definición de una ruta *default,* (es decir, por donde el tráfico será enviado si no se especifica otra ruta) y cuando el equipo lo permite, la configuración de la política de ancho de banda. En este punto, se ejecutan comandos de obtención de la configuración del mismo. Este se identifica con el código

GET_CONFIG. Adicionalmente, se puede ejecutar un comando para determinar la respuesta de la IP del CPE desde el sistema local, esto se identifica como PING_LOCAL y también se hace con la herramienta *ping*.

Cada dispositivo y medio de conexión puede ser un punto de falla. Para llegar a un diagnóstico se deben ejecutar los comandos de cada dispositivo; analizar sus respuestas y luego aplicar un conjunto de reglas que determinarán el diagnóstico del servicio. Un ejemplo de las conexiones de los dispositivos configurados en un servicio de internet se puede observar en la figura 15. Aquí el PE se llama gtriara, el *switch* se llama mtriara y el CPE tiene una IP WAN 10.160.64.153.

Figura 15. Diagrama de un servicio de acceso a internet. Tomado de la empresa de telecomunicaciones.

Los lugares para la ejecución de comandos son: localmente (donde se encuentre el sistema), en el *router* PE, en el *switch* de acceso y en el *router* CPE. Los comandos para las pruebas en equipos Cisco se presentan en la figura 16.

Localización	Código	Nombre	Comando
CPE	GET_CONFIG	Obtener Configuración	show running-config
Local	PE_DOWN	Nodo disponible	ssh -l user@ipNodo
Local	SWT_DOWN	Switch disponible	ssh -l user@ipSwitch
Local	LOCAL_PING	Ping a dispositivo	ping -c5 ipDispositivo
PE	IP_WAN_OK	IP WAN Arriba	ping ipWan
PE	IP_GEST_OK	IP GESTION Arriba	ping ipGestion
Switch	PORT_REGISTERED	Puerto Switch Registrado	show running interface GiXX
Switch	REGISTERING_MAC	Puerto Switch registrando MAC	show mac-address-table GiXX
Switch	OPTICAL_POWER	Potencia óptica	show pon power onu-rx gpon-onu_W/X/Y:Z
Switch	OLT_PORT_OK	Puerto OLT caido	show interface gpon-olt_W/X/Y
CPE	PORT_CPE_UP	Puerto en CPE operativo y administrativo en up	show interface GiXX
Switch	VLAN_CONFIG	VLAN configurada	show running interface GiXX

Figura 16. Tabla de comandos a ejecutar en los dispositivos. Creado por el autor a partir de la información suministrada por la empresa de telecomunicaciones

La información de los diagnósticos, se presenta en la figura 17. Aquí se pueden apreciar los diagnósticos más comunes y obtenibles por medio de la ejecución de comandos sobre los elementos de red. Además del diagnóstico, se presenta la acción a seguir como una medida para solucionar el incidente.

Diagnóstico	Código	Localización	Acción
Ultima milla OK	SERVICE_OK	CPE	Verificar conectividad LAN en el lado del cliente
Bloqueo	CPE_BLOCKED	CPE	Reiniciar CPE y TRV
Error Configuración	CPE_CONFIG_ERR	CPE	Confirmar topología LAN
Falla Fibra	FIBER_FAIL	Medio	Escalar con fallo de fibra/Puertos OLT
Falla Nodo	PE_FAIL	PE	Escalar por caida de equipo
Tabla ARP vacía	SWT_ARP_FAIL	Switch	Reiniciar CPE y TRV
Interfaz caida	SWT_INT_DOWN	Switch	Verificar energia y conectividad. Reiniciar CPE y TRV
Error Configuración Puerto	SWT_PORT_CONFIG	Switch	Escalar
Falla Switch	SWT_FAIL	Switch	Escalar por caida de equipo

Figura 17. Diagnósticos a realizar por el sistema. Creado por el autor a partir de la información suministrada por la empresa de telecomunicaciones

Los diagnósticos son obtenidos aplicando reglas a los resultados de la ejecución de comandos. Estas reglas fueron suministradas por los ingenieros del NOC y reflejan su actividad diaria; por ejemplo, si para un servicio, a nivel de puerto de *switch* se están registrando direcciones MAC y la interfaz de gestión no responde a ping, se cumple la regla:

CPE_BLOCKED:=REGISTERING_MAC & !(GEST_OK)

donde a la derecha se observa el código del diagnóstico y a la izquierda, las dos condiciones de registro de MAC y respuesta de la IP de gestión, esta última negada (!), es decir, la IP WAN del CPE no está respondiendo cuando se utiliza la herramienta *ping*, pero en el switch de acceso si aparecen registradas direcciones MAC del cliente. Para solucionar esta falla, se requiere el reinicio del CPE.

En caso de que el *router* PE se encuentre apagado o no responda, el diagnóstico será PE_FAIL. Algo similar para el caso en que el *switch* de acceso se encuentre apagado o no responda, el diagnóstico será SWT_FAIL. Para determinar si el servicio está funcionando correctamente, es necesario que el puerto en el *switch* esté registrado (PORT_REGISTERED), que el *switch* en la interfaz del servicio posea una tabla de direcciones MAC con los dispositivos del lado del cliente (REGISTERING_MAC) y que la interfaz de gestión responda al comando *ping* desde el nodo (IP_GEST_OK). Un resumen de estas reglas se observa en la figura 18.

Diagnóstico	:=	Regla
SERVICE_OK	:=	PORT_REGISTERED && REGISTERING_MAC && IP_GEST_OK
CPE_BLOCKED	:=	REGISTERING_MAC && ! IP_GEST_OK
CPE_CONFIG_ERR	:=	! GET_CONFIG
FIBER_FAIL	:=	! PORT_REGISTERED && ! PORT_CPE_UP
PE_FAIL	:=	PE_DOWN
SWT_ARP_FAIL	:=	! REGISTERING_MAC
SWT_INT_DOWN	:=	! PORT_REGISTERED
SWT_PORT_CONFIG	:=	TRUNK_MODE || ! VLAN_CONFIG
SWT_FAIL	:=	SWITCH_DOWN

Figura 18. Reglas para encontrar los diagnósticos. Creado por el autor a partir de la información suministrada por la empresa de telecomunicaciones

En este capítulo se determinaron los diagnósticos y las reglas a utilizar en el sistema a desarrollar. En el próximo capítulo se presentan los requerimientos del sistema.

4. Requerimientos del sistema multi-agente a desarrollar

Según la ingeniería de software, en Pressman, R. S. [63], los requerimientos de un sistema describen los servicios que ha de ofrecer el sistema y las restricciones asociadas a su funcionamiento. Entre los requerimientos se encuentran los funcionales y los no funcionales. Los requerimientos funcionales expresan la naturaleza del funcionamiento del sistema (cómo interacciona el sistema con su entorno y cuáles van a ser sus estados y su funcionamiento). Los requerimientos no funcionales son restricciones sobre el espacio de soluciones. Se diferencian en que, mientras los funcionales indican qué debe hacer el sistema, los no funcionales especifican cómo hacerlo.

En este capítulo se presentan los requerimientos del sistema multi-agente. Se incluyen, además, aspectos como la perspectiva del sistema, la descripción funcional y las restricciones de diseño e implementación.

4.1 Perspectiva del sistema

El sistema surge como una solución a problemas tales como: que el diagnóstico de fallas de servicios de internet es demorado, que es proclive a errores humanos y que requiere conocimiento técnico de varios dispositivos.

El sistema a desarrollar debe automatizar el diagnóstico y facilitar que personal sin conocimiento técnico profundo, apoyado en el sistema pueda realizar diagnósticos acertados.

El sistema multi-agente (MAS, por su sigla en inglés) debe interactuar con el ingeniero del centro de llamadas *(callcenter)*, para recibir peticiones de diagnóstico y retornarlas atendidas, con la red de dispositivos MPLS (*Multiprotocol Label Switching*) para la ejecución de los comandos, con la base de datos de configuraciones CMDB (*Configuration Management Data Base*) para obtener información de los servicios y con el coordinador de NOC para entregar reportes de uso.

El sistema debe contar con una interfaz web para que los usuarios interactúen con él, una infraestructura de servicios web (*web services*) para conectar los diferentes componentes y debe estar soportado por un MAS. La representación gráfica del sistema y sus interacciones se puede observar en la figura 19.

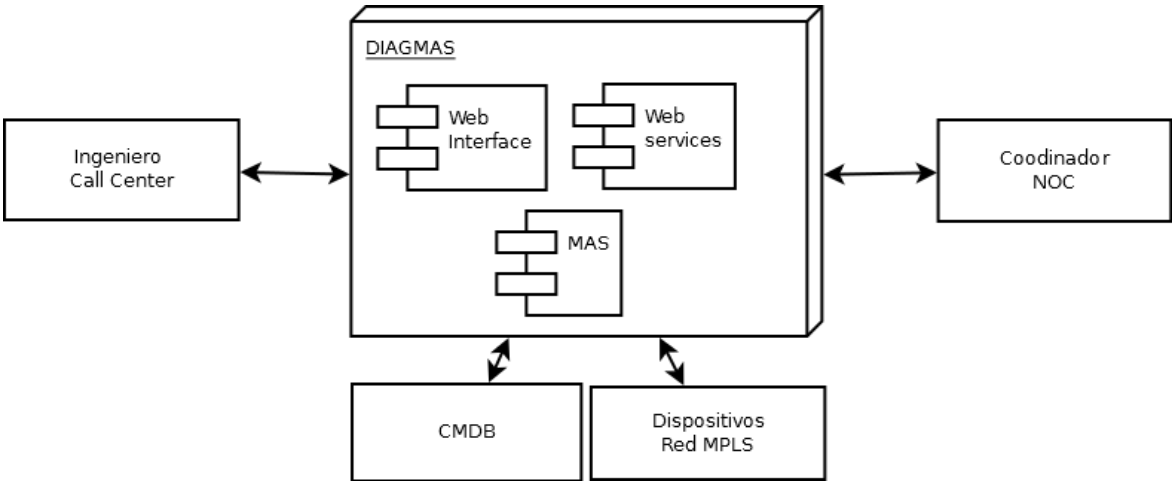

Figura 19. Perspectiva del sistema. Creado por el autor.

4.2 Funciones del sistema

Según la ingeniería de software en Pressman, R. S. [63], las funciones del sistema expresan la naturaleza del funcionamiento del sistema, es decir, cómo interactúa el sistema con su entorno y cuáles van a ser sus estados y funcionamiento. Para el sistema a desarrollar se identificaron las siguientes funciones:

Función 1: Diagnosticar un servicio, con el fin de automatizar este proceso y hacer más rápida la atención de incidentes. El criterio de aceptación será entregar diagnósticos equivalentes a los de los expertos humanos en los casos de prueba definidos.

Función 2: Obtener información de los servicios de la CMDB corporativa, para poder ejecutar los comandos con en los dispositivos que correspondan al servicio. El criterio de aceptación es que se encuentre información de los servicios de pruebas.

Función 3: Ejecutar comandos sobre los dispositivos asociados al servicio, con el fin de tener la entrada para la realización del diagnóstico. El criterio de aceptación es la

ejecución de los comandos definidos por los ingenieros NOC y obtener los mismos resultados que cuando son ejecutados por ellos.

Función 4: Interpretar los resultados de la ejecución de comandos, con el fin de determinar los valores para aplicar las reglas. El criterio de aceptación serán pruebas unitarias.

Función 5: Aplicar el conjunto de reglas de decisión para determinar el diagnóstico. El criterio de aceptación serán pruebas unitarias.

Función 6: Entregar reporte de uso, con el fin de que el coordinador de NOC pueda hacer análisis de los casos más repetitivos. El criterio de aceptación será la validación por el usuario.

4.3 Clases de usuarios y características

El sistema debe contar con dos clases de usuarios, los ingenieros *callcenter* y el coordinador NOC. Las características de los usuarios son:

Ingeniero *callcenter*: encargado de registrar el incidente ya sea por una llamada del cliente o un medio automático. Cuenta con una limitada experiencia en el diagnóstico de incidentes. Es el usuario principal del sistema.

Coordinador NOC: facilitador y orientador de las labores de los ingenieros de *callcenter*. Revisa el uso del sistema y funge como líder técnico del mismo. Determina posibles cambios o ajustes.

4.4 Ambiente de operación

Según NetworkMarketShare, sitio web de estadísticas de distribución del mercado para tecnologías de internet [51], Microsoft Windows con sus versiones XP, 7 y 10 es el sistema operativo que cuenta con el mayor participación del mercado, por lo tanto, el sistema a desarrollar debe funcionar en este sistema operativo.

A nivel de servidor, debido al tipo de licenciamiento, se recomienda utilizar servidores linux tanto para los contenedores de agentes como para la aplicación web.

4.5 Restricciones de diseño e implementación

El sistema debe conectarse a enrutadores, ya sean PE (enrutadores de borde) o CPE (enrutadores del lado del cliente). La conexión debe hacerse usando el protocolo ssh (*secure shell*), estableciendo una sesión en la CLI (*Command Line Interface*) y realizando únicamente comandos de lectura.

En cuanto al *switch* de acceso, el sistema debe tener la capacidad de conectarse a equipos usando ssh y ejecutar comandos en ellos a través de la CLI.

Para las tareas de conexión a dispositivos, administración de la conexión a los dispositivos, evaluación de reglas y generación de reportes, el sistema deberá usar componentes de código libre o con licencias GNU, GPL o Apache Commons.

Según el portal de estadísticas Statista [64], el mayor segmento de mercado en *switchs,* lo tiene la compañía Cisco [65]. Por esto, por recomendación de los expertos humanos y porque se pueden simular usando GSN3 en un computador personal, en el presente proyecto, los módulos de comunicación a los dispositivos deben ser desarrollados para dispositivos de esta marca.

4.6 Supuestos y dependencias

Se supone que el sistema será utilizado por personal con conocimientos técnicos limitados, por lo que debe ser amigable al usuario.

Se supone que el sistema será actualizado en el tiempo por los ingenieros del NOC, por lo que se debe documentar la forma de incluir nuevas plataformas y comandos a ejecutar.

El sistema es interdependiente con la CMDB y la red MPLS, por lo que cualquier cambio en uno u otro debe ser conocido, determinado su impacto y realizar la adaptación del sistema a éste.

4.7 Características del sistema

Requerimiento 1. Búsqueda de la información de los dispositivos e interfaces asociadas al servicio.

Requerimiento 2. Ejecución de comandos sobre los dispositivos del servicio. Estos comandos son consultas requeridas para el diagnóstico.

Requerimiento 3. Interpretación de los resultados de la ejecución de comandos.

Requerimiento 4. Aplicación de reglas para determinar el diagnóstico del servicio usando la interpretación de la ejecución de comandos.

En este capítulo se describieron los requerimientos para el sistema; en el próximo se presenta el análisis y del diseño del mismo.

5. Análisis y diseño del sistema multi-agente

En este capítulo se presentan los resultados de seguir la metodología de análisis y diseño de sistemas multi-agente presentada en Nikraz, M. et al. [66].
Se inicia con la descripción de la metodología y herramientas empleadas y finaliza con los entregables, entre ellos el listado de agentes, responsabilidades, mensajes e interacciones.

5.1 Selección de la herramienta y metodología

Se seleccionó la herramienta *Java Agent DEvelompment Framework* (JADE) y la metodología de desarrollo propuesta por Nikraz et al. [66], basándose en la experiencia previa del grupo de investigación LISI en el campo de MAS, así como en los trabajos de grado que utilizaron MAS.

El proceso de selección se presenta en la tesis de Ardila, E. [4]. Para éste, se tuvieron en cuenta, los criterios de: lenguaje de programación de fácil aprendizaje, sistema operativo de la solución disponible, licenciamiento económico, objetivos de la herramienta, recursos y soporte disponible para la misma.

Las herramientas revisadas, entre otras, fueron: ABLE (Agent Building and Learning Environment), ADK (Tryllian Agent Development Kit), AgentBuilder, AgentSheets, Altreva Adaptive Modeler, AnyLogic, AOR simulation, Ascape, Brahms, Breve, Construct, Cormas, Cougaar, D-OMAR(Distributed Operator Model Architecture), DeX, ECHO, ECJ (Evolutionary computation), FAMOJA (Framework for Agent-based MOdeling with java), Framsticks, GPU Agents, GROWlab, ICARO-T, iGen, JADE, JAMEL, janus, JAS, JASA (Java Aution Simulator API), JCA-Sim, jEcho, iES (Java enterprise Simulator), JESS, JIAC, LSD (Laboratory for Simulation Development), MacStarLogo, Madkit, MAGSY, MAML, MAS-SOC, MASON, MASS, MIMOSE, Moduleco, MOOSE, Netlogo, OBEUS (Object Based Environment for Urban Simulation), Omonia, OpenStarLogo, oRIS.

Basándose en Ardila, E. [4], la herramienta JADE fue la seleccionada debido a cumple los estándares FIPA, a que su lenguaje de programación es Java, a que para su despliegue, es una solución multiplataforma que sólo requiere una Java Virtual Machine en la máquina donde se vaya a desplegar, también porque su licenciamiento es LGPL (*Lesser General Public License Version 2*), además porque es mantenida por Telecom Italia y por último porque es software libre.

Respecto a su propósito, el de construir aplicaciones distribuidas compuestas de entidades autónomas, se alinea con los requerimientos del presente proyecto.

Así mismo, basándose en Ardila, E. aprovechando la experiencia de los trabajos previos del grupo de investigación LISI, se escogió la metodología de análisis y diseño de MAS propuesta por Nikraz et al. utilizando JADE [66].

5.2 Metodología y herramientas empleadas

La metodología propuesta por Nikraz et al. [66] presenta las fases de planeación, análisis, diseño, implementación y pruebas de manera similar a otras, pero limitadas por la funcionalidad de la herramienta JADE.

Según Bellifemine, F. L., [13], JADE es un middleware que simplifica la implementación de sistemas MAS proporcionando una interfaz gráfica que soporta las fases de desarrollo y depuración, así como las operaciones de configuración. También contiene una plataforma multi-agente que puede estar distribuida a través de múltiples máquinas con diferentes sistemas operativos.

El diagrama de flujo de la metodología para desarrollo de sistemas multi-agente de Nikraz et al. [66] se presenta en la figura 20. En ella se pueden apreciar los pasos asociados a

cada fase con sus entregables y la secuencia de los mismos.

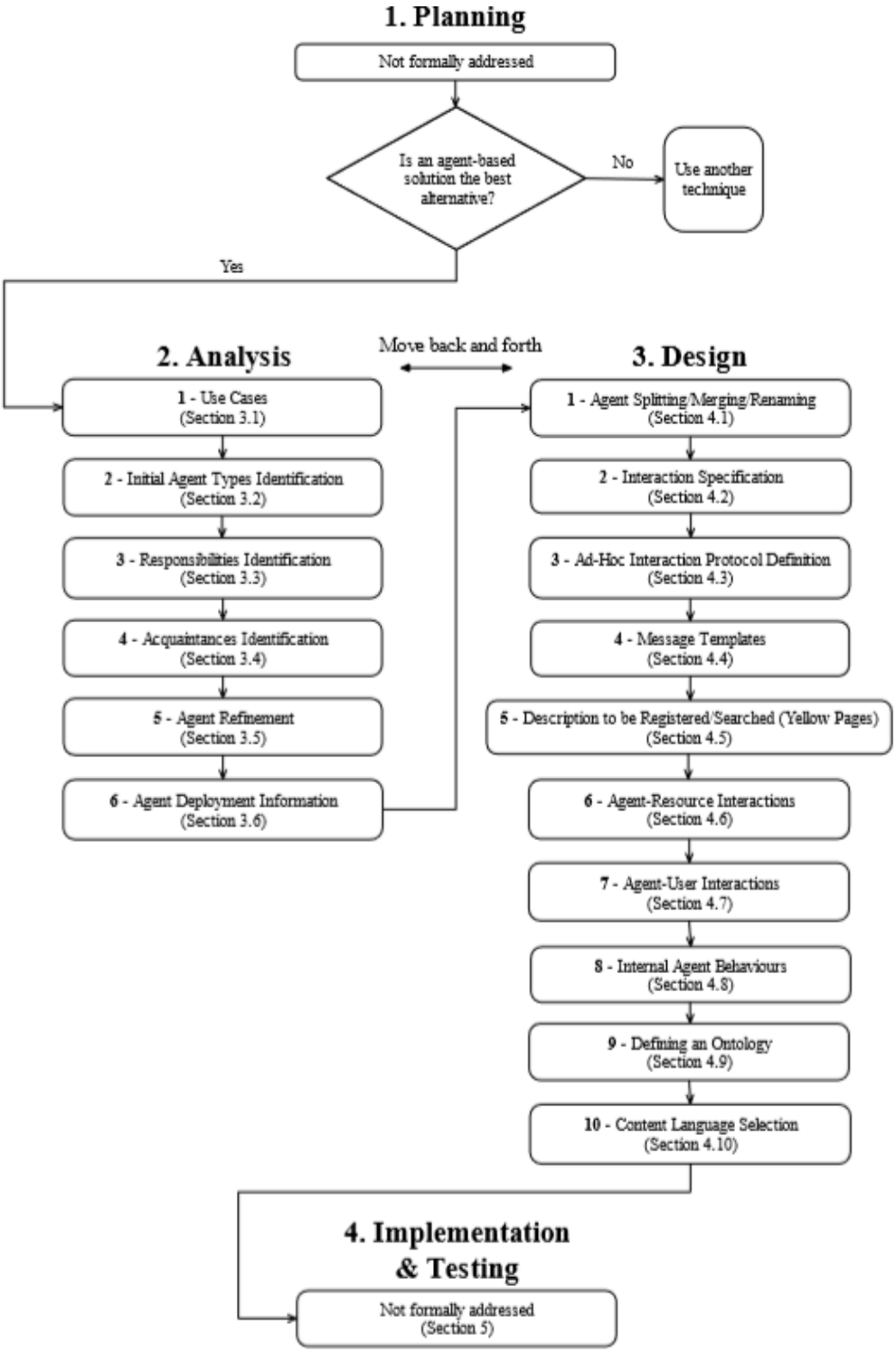

Figura 20. Metodología de análisis y diseño de MAS. Tomado de Nikraz, M. et al. **[66]**.

5.3 Planeación

La fase de planeación, así como la implementación y las pruebas, no son formalmente tratadas por la metodología de Nikraz, M. et al. [66]. Sin embargo, se incluye en una decisión en el diagrama de flujo de la metodología, la cual implica que el diseñador haya tomado una decisión racional de usar o no una solución basada en agentes [66]. Si la respuesta es sí, el diseñador, continua a la fase de análisis, en caso contrario, debe buscar una solución alternativa. Nikraz, M. et al. [66], afirman que para aplicar su metodología, el único supuesto requerido como producto de la planeación es que se haya escogido una solución basada en agentes como la mejor alternativa.

En el presente trabajo, como alternativas al uso de un MAS se presentan los sistemas de monitoreo de alarmas y detección de comportamiento atípico (*aberrant behavior*) así como los módulos de gestión de fallas de sistemas OSS (*Operational Support Systems*).

Estas dos alternativas permiten determinar problemas masivos de la red, sin embargo, están orientadas a plataformas de una sola marca. La solución propuesta llena el vacío de facilitar un solo punto de vista de los equipos de diferentes marcas que contribuyan a la prestación de un servicio para un cliente particular. Esto permitirá llegar a una solución más rápida para las fallas que puedan presentarse.

5.4 Análisis

En la etapa de análisis se pretende modelar la solución al problema usando agentes. Según la metodología de Nikraz et al. [66], debe incluir la definición de casos de uso, de agentes y sus responsabilidades, interacciones y despliegue. A continuación se presentan los pasos indicados por la metodología y los resultados de aplicarlos para el análisis del sistema objeto de este trabajo.

5.4.1 Casos de Uso

EL primer paso es definir los casos de uso, pues éstos permiten determinar los potenciales requerimientos funcionales del sistema. En la figura 21, se puede apreciar como el ingeniero *callcenter* hace un diagnóstico de un servicio, y para ello hace uso de los casos de consulta en la CMDB, ejecución de pruebas, análisis de las mismas, selección de diagnóstico y registro de operaciones. A su vez, el coordinador NOC, genera reportes del uso del sistema y para ello usa el caso de consulta de históricos.

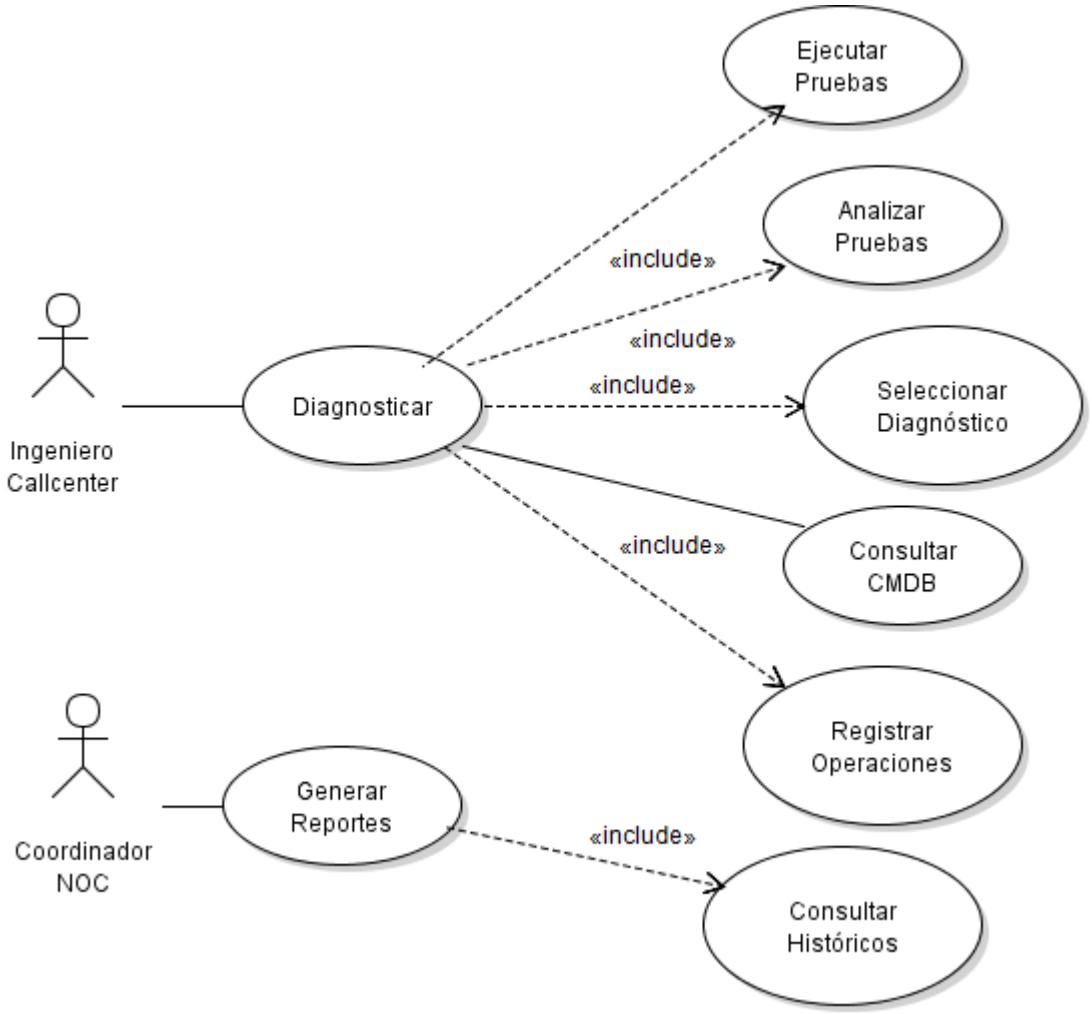

Figura 21. Casos de uso del sistema. Creado por el autor

5.4.2 Identificación inicial de los tipos de agentes

En el segundo paso de la metodología presentada por Nikraz, M. et al, [66], se identifican los agentes asociados a usuarios o dispositivos, así como agentes por cada uno de los recursos utilizados. En la figura 22 se puede apreciar un agente, representado como un óvalo, para cada actor (ingeniero *callcenter*, en este caso, se crea el agente *callcenter*), para cada recurso (bases de datos de servicios y diagnósticos, se crean los agentes *database* y *diagnosticator*) y para cada dispositivo (se crea el agente *device*).

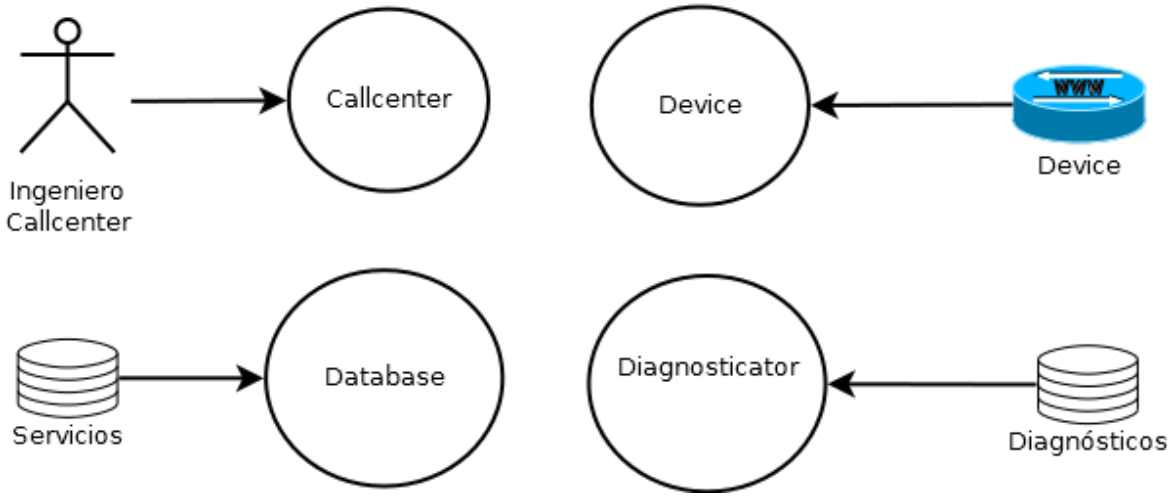

Figura 22. Identificación de agentes. Creado por el autor.

5.4.3 Identificación de responsabilidades

En el tercer paso de la metodología presentada por Nikraz, M. et al. [66], para cada tipo de agente identificado se hace una lista de sus principales responsabilidades. Estas se deben derivar de los casos de uso definidos anteriormente y se deben considerar primero los agentes cuyas responsabilidades sean más claras. En la tabla 2, se presentan los agentes y las responsabilidades identificadas inicialmente.

Tipo de agente	Responsabilidades
Callcenter	Servir las peticiones de diagnóstico del ingeniero *callcenter*.
Device	Conectarse a un dispositivo y ejecutar comandos para obtener información sobre el estado de un servicio configurado en éste.
Diagnosticator	Reunir los resultados de los comandos ejecutados. Llegar a un diagnóstico partiendo de los resultados reunidos de la ejecución de comandos.
Database	Entregar información del servicio.

Tabla 2. Identificación de responsabilidades de los agentes.

5.4.4 Identificación de las interacciones

En el cuarto paso de la metodología presentada por Nikraz, M. et al. [66], se deben encontrar las interacciones que se presentan entre los agentes. En la figura 23 se observa como el *callcenter* agent interactúa con el *database*, también como el agente *diagnosticator* interactúa con el *device* y, a su vez, con el *callcenter*. Estas interacciones encontradas, deben adicionarse a la tabla de responsabilidades, en la figura 24, se puede apreciar la nueva tabla.

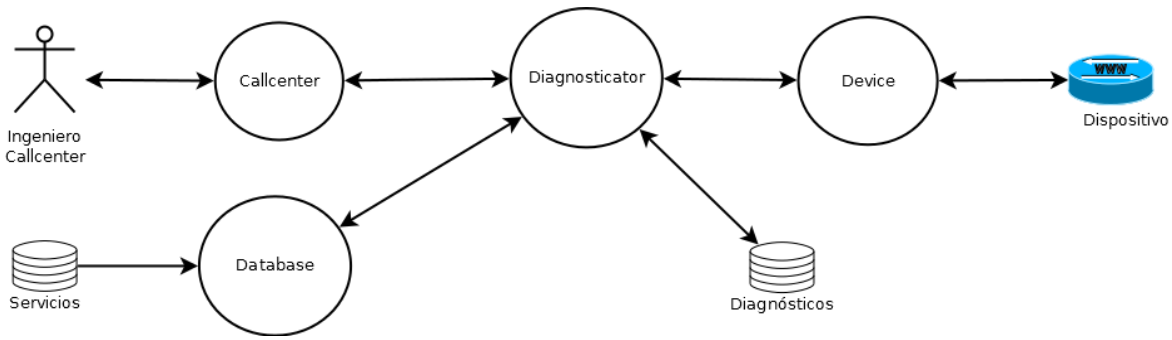

Figura 23. Diagrama de agentes con sus interacciones. Creado por el autor.

Tipo de agente	Responsabilidades
Callcenter	Servir las peticiones de diagnóstico del ingeniero *callcenter*. Consultar información del servicio al agente *database*. Solicitar al agente *diagnosticator* un diagnóstico. Informar al usuario la respuesta del agente.
Device	Conectarse a un dispositivo y ejecutar comandos para obtener información del estado de un servicio configurado en éste. Responder a peticiones de ejecución de comandos.
Diagnosticator	Recibir información sobre el servicio del agente *callcenter*. Recibir la petición de diagnóstico de parte del agente *callcenter*. Entregar el diagnóstico al agente *callcenter*. Solicitar pruebas de los agentes *device* relacionados con el servicio. Reunir los resultados de la ejecución de comandos. Llegar a un diagnóstico partiendo de los resultados reunidos de la ejecución de comandos.
Database	Entregar información del servicio. Recibir petición del agente *callcenter.* Enviar respuesta al agente *callcenter.*

Tabla 3. Tabla de responsabilidades luego de identificar las interacciones entre agentes.

5.4.5 Refinamiento de agentes

En el quinto paso de la metodología presentada por Nikraz, M. et al. [66], se refina el conjunto de agentes aplicando las consideraciones de soporte, descubrimiento y administración. Soporte se refiere a cuál información requieren para cumplir sus responsabilidades, descubrimiento se refiere a cómo se van a buscar y encontrar los agentes entre sí y administración a cómo se mantiene un registro de los agentes existentes y sus procesos de inicio y terminación.

En la figura 24, se observan nuevos agentes como el DF (*Directory Facilitator*), AMS (*Agent Management System*), RMA (*Remote Monitoring Agent*), provistos por JADE y el agente *network* encargado de mantener los agentes *device*.

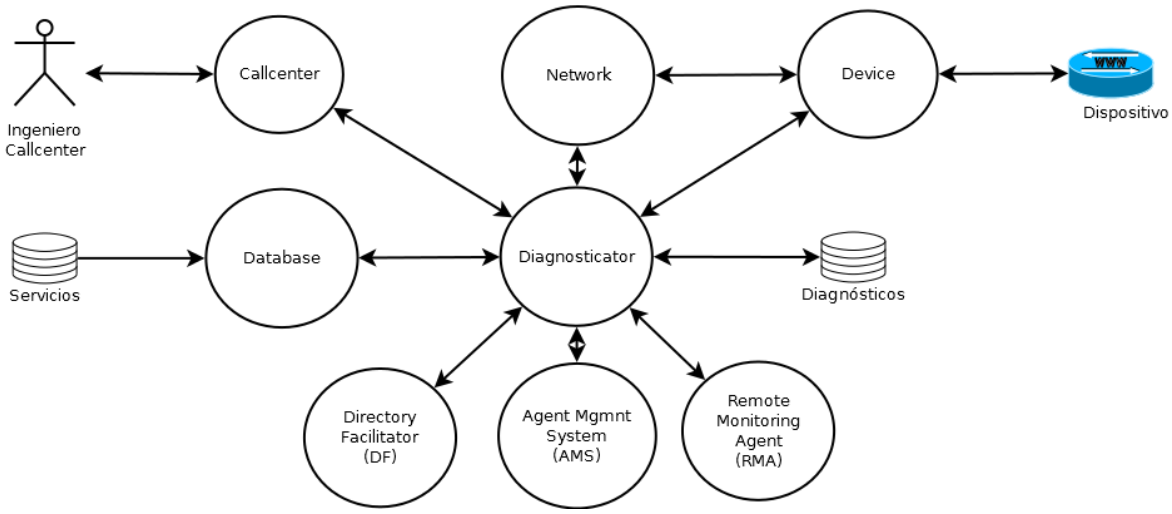

Figura 24. Diagrama de agentes luego del paso de refinamiento.

Tipo de agente	Responsabilidades
Callcenter	1. Recibir las peticiones de diagnóstico de un servicio enviadas por el ingeniero *callcenter*. 2. Consultar información del servicio al agente *database*. 3. Consultar al *Directory Facilitator* los agentes *device* relacionados a un servicio en particular. 4. Solicitar al agente *diagnosticator* un diagnóstico de un

	servicio. 5. Recibir el diagnóstico. 6. Responder al usuario la petición de diagnóstico.
Device	7. Conectarse a un dispositivo y obtener información sobre el estado de un servicio configurado en éste. 8. Responder a peticiones de ejecución de comandos. 9. Registrarse en el agente *Directory Facilitator*.
Diagnosticator	10. Reunir los resultados de la ejecución de comandos. 11. Llegar a un diagnóstico partiendo de los resultados reunidos de la ejecución de comandos. 12. Recibir información sobre el servicio del agente *callcenter*. 13. Recibir la petición de diagnóstico de parte del agente *callcenter.* 14. Entregar el diagnóstico al agente *callcenter.* 15. Registrarse en el *Directory Facilitator*. 16. Solicitar ejecutar comandos a los agentes *device* relacionados con el servicio. 17. Recibir resultados de la ejecución de comandos.
Network	18. Crear los agentes *device*.
Database	19. Entregar información del servicio. 20. Recibir petición de información de un servicio. 21. Responder petición de información de un servicio.
Directory Facilitator (DF)	22. Servir de páginas amarillas para permitir buscar y encontrar agentes existentes.
Agent Management System (AMS)	23. Dar nombres a los agentes
Remote Monitoring Agent	24. Presentar gráficamente el estado de la plataforma de agente.

(RMA)	

Tabla 4. Agentes y responsabilidades, después del paso de refinamiento.

5.4.6 Información de despliegue de agentes

En el sexto paso de la metodología presentada por Nikraz, M. et al. [66], se define como se hará el despliegue de los agentes. Se crea un diagrama de despliegue de agentes con los equipos donde van a ser desplegados los agentes. En la figura 25 se puede apreciar un servidor donde está el contenedor de agentes y como un servidor remoto puede tener un contenedor de agentes también.

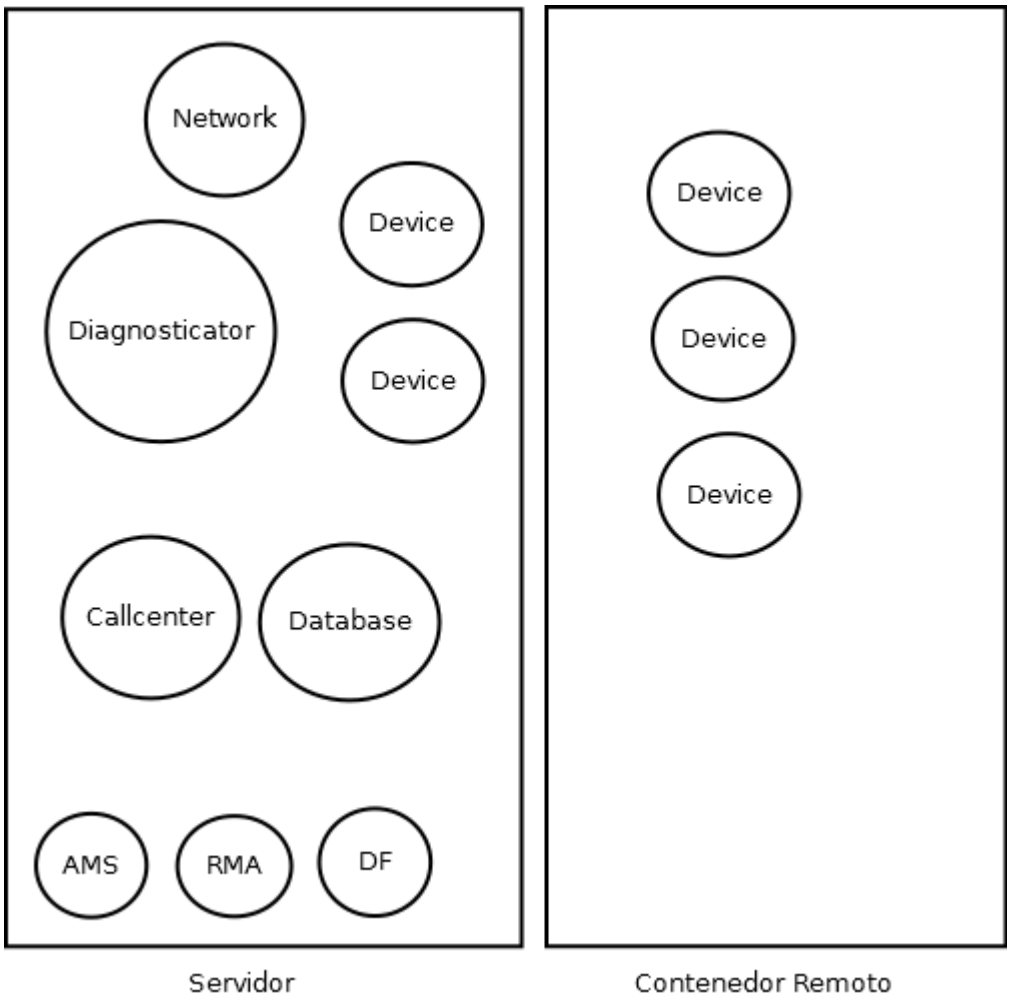

Figura 25. Despliegue de agentes.

5.5 Diseño

Según la ingeniería de software, Pressman, R.S., [63], en esta etapa se busca especificar la solución, de tal forma que la transición a la implementación sea más sencilla, con la posibilidad de generar código automáticamente. ´Según la metodología propuesta por Nikraz, M. et al. [66], abarca pasos para definir interacciones, protocolos y estructura de mensajes.

5.5.1 División o unión de agentes

En el primer paso según Nikraz, M. et al. [66], se determina si cada agente debe ser dividido en dos o más agentes, o si debe ser unido a otro agente. Algunas reglas de decisión son: evitar la duplicidad de datos; evitar la duplicación de código y balancear entre la cantidad de agentes sencillos y la complejidad de los mismos. Para el actual trabajo los agentes definidos hasta el momento se conservan sin divisiones o uniones.

5.5.2 Especificación de interacción

En el segundo paso, según Nikraz, M. et al. [66], se toman en cuenta todas las responsabilidades relacionadas con las interacciones y se produce una tabla de interacciones para cada tipo de agente.

Cada fila en la tabla representa una interacción que contiene: nombre descriptivo de la interacción, número de la responsabilidad, protocolo de la interacción, *role* (Iniciador, Respuesta), tipo de agente del rol complementario y condición de inicio de la interacción. La especificación de las interacciones entre los agentes del sistema desarrollado, se puede observar en la tabla 5.

Nombre	Resp	Protocolo	Role	Con	Cuando
Petición de diagnóstico de servicio	4	FIPArequest	I	*Diagnosticator*	El ingeniero *callcenter* pide diagnosticar un servicio.
Responder a la petición de diagnóstico del servicio	6	FIPArequest	R	*Diagnosticator*	El agente *diagnosticator* entrega el diagnóstico.
Buscar los agentes asociados al servicio	22	FIPArequest	I	*Directory Facilitator*	Desde el inicio del sistema
Petición de ejecución de comandos	16	FIPArequest	I	*Device*	Petición de diagnóstico de servicio
Recibir resultados de la ejecución de comandos.	17	FIPArequest	R	*Device*	Comandos ejecutados
Recibir petición información de servicio	20	FIPArequest	I	*Diagnosticator*	Petición de información de servicio.
Responder petición de información de servicio	21	FIPArequest	R	*Diagnosticator*	Respuesta de la base de datos.

Tabla 5. Especificación de interacciones entre agentes.

5.5.3 Definición del protocolo de interacción ad-hoc

El tercer paso, según Nikraz, M. et al. [66], consiste en la definición de protocolos adicionales para las interacciones. La recomendación es utilizar los protocolos de interacción definidos por la FIPA [18]. En caso de que para las interacciones a implementar, estos protocolos sean insuficientes, se requiere definir un nuevo protocolo.

Para el sistema desarrollado, las interacciones entre los agentes serán peticiones (*request*), es decir, solicitudes de un agente a otro para que realice alguna acción. Para ello, se escogió el protocolo FIPA request, no existiendo otros tipos de interacciones, no se requieren más protocolos.

5.5.4 Estructura de mensajes

El cuarto paso, según Nikraz, M. et al. [66], se identifican y adicionan a la tabla de interacción los posibles objetos mensaje intercambiados entre los agentes. Todas las interacciones de los roles identificadas en los pasos anteriores se implementan como comportamientos de JADE. Adicionalmente, mezclar en una estructura aquellos mensajes que tengan la misma combinación de inicio y lenguaje.
Para el sistema desarrollado, en la tabla 6, se puede observar la estructura de los mensajes asociados a cada interacción.

Interacción	Resp.	Protocolo	Role	Con	Cuando	Estructura
Petición de diagnóstico de servicio	4	FIPArequest	I	*Callcenter*	El usuario reporta un incidente	Id-conversación código-servicio

Responder a la petición de diagnóstico del servicio	6	FIPArequest	R	*Callcenter*	Agente diagnosticator entrega diagnóstico	id-conversación código-servicio código-diagnóstico log-comandos
Buscar los agentes asociados al servicio	22	FIPArequest	I	Directory Facilitator	Inicio del sistema	nombre-agente
Petición de ejecución de pruebas	16	FIPArequest	I	Diagnosticator	Petición de diagnóstico de servicio	id-conversación nombre-agente comando-a-ejecutar
Recibir resultados	17	FIPArequest	R	Diagnosticator	Datos capturados	id-conversación nombre-agente resultado-ejecución
Petición de información	20	FIPArequ	I	Diagnostic	Petición de	id-

de servicio		est		ator	diagnóstico	converación código-servicio
Responder petición de información de servicio	21	FIPArequest	R	Diagnostic ator	Diagnóstico completo	id-conversación código-servicio dispositivo-PE dispositivo-Switch dispositivo-CPE

Tabla 6. Estructura de los mensajes.

5.5.5 Páginas amarillas de agentes

En el quinto paso, según Nikraz, M. et al. [66], se formalizan las convenciones de nombres para los agentes registrados/buscados en el catálogo de páginas amarillas de agentes gestionado por el agente JADE *Directory Facilitator*. Para el sistema desarrollado, los agentes a buscar son los asociados a dispositivos (*device*). Estos deberán registrarse en el catálogo de páginas amarillas de JADE con un identificador único que será la dirección IP asociada al dispositivo.

5.5.6 Interacciones agente-recurso

En el sexto paso, según Nikraz, M. et al. [66], se determina la forma de realizar las interacciones entre agentes y recursos externos. Para que el sistema desarrollado se comunique con recursos externos como dispositivos de la red y bases de datos, se

utilizan servicios web (*web services*). El agente hace un llamado al servicio web apropiado y este le retorna una respuesta a su petición.
Los servicios web a consumir deben ser:
1. Servicio web de ejecución de comandos.
2. Servicio web de consulta de información de los servicios de cliente.

5.5.7 Interacciones agente-usuario

En el séptimo paso, según Nikraz, M. et al. [66], se deben definir cómo se realizarán las interacciones entre los agentes y los usuarios. Enfocándose en la interfaz de usuario gráfica o GUI (*Graphic User Interface*), JADE permite dos opciones, una GUI local, implementada en Java *Swing* y una GUI web, implementada en *JavaServer Pages.*

En el presente trabajo, el usuario tendrá interacciones únicamente con el agente *callcenter*, quien le entregará la respuesta a la petición de diagnóstico. El envío de peticiones y el manejo de las respuestas requiere el diseño de una interfaz gráfica independiente de JADE, debido a que el ambiente proporcionado por JADE es útil en el desarrollo y la depuración del sistema, pero no es lo suficientemente fácil de manejar para un usuario sin conocimientos técnicos en sistemas multi-agente. Esta interfaz gráfica, se diseña como una aplicación web. Lo que evita la instalación de software adicional del lado del usuario del sistema.

5.5.8 Comportamientos internos del agente

En el octavo paso, según Nikraz, M. et al. [66], se define el comportamiento de cada agente. Las tareas que realiza un agente se implementan como comportamientos de la clase Agent. Se debe pasar de las responsabilidades a comportamientos de agente. Se debe tener en cuenta que para cada responsabilidad en la tabla de responsabilidades, se debe encontrar una clase cuya funcionalidad se debe extender y definir en ella este comportamiento. Esto depende del protocolo de interacción seleccionado.

En el presente trabajo cada agente tiene estados de recepción de peticiones y de respuesta a las mismas. Los comportamientos adicionales de cada agente se determinan por las responsabilidades asociadas a cada uno, por ejemplo, el agente *device*, además debe enviar peticiones de ejecución de comandos en los dispositivos y recibir las respuestas de éstos.

5.5.9 Definición de una ontología

En el noveno paso, según Nikraz, M. et al. [66], se define la ontología a usar en el sistema. En la metodología seguida, una ontología se entiende como un conjunto de conceptos, predicados y acciones de agente referidos a un dominio dado. Los conceptos son tipos complejos de entidades abstractas o concretas que existen en el ambiente en que reside el agente. Estas se especifican con un nombre y un conjunto de valores de tipos de datos predefinidos, a los campos donde se guardan los valores se les denomina casillas (*slots*).

En el presente trabajo se utilizaron las definiciones del glosario de ATIS Telecom, mantenido por el ANSI (*American National Standards Institute*) [67] y se determinaron como conceptos: el incidente (evento con potenciales efectos adversos en el sistema), el servicio (una funcionalidad provista por un proveedor a sus clientes), los puertos (punto donde las variables de la red pueden ser observadas o medidas, interfaces (en los dispositivos *router*, *switch* y CPE), los diagnósticos (causa o naturaleza del incidente dentro de los elementos específicos del sistema), prueba (medidas físicas a los equipos de red), variable (representación de una medida física en un equipo) y resultado (conjunto de estados resultado-positivo, resultado-negativo, no-determinado)

Según Nikraz, M. et al. [66], los predicados son estructuras de relaciones especificadas en términos de un nombre y un conjunto de *slots* cuyos valores con tipos predefinidos. Para decidir si se utiliza un predicado o un *slot* dentro de una entidad para representar la asociación de dos entidades, se usa la regla de que si la entidad dependiente cambia durante la vida del sistema, se debe utilizar un predicado. Adicionalmente, se deben determinar las acciones que puedan producir un efecto o generar un resultado a ser enviado al que realiza la petición.

En la figura 26, se pueden apreciar los conceptos, en la figura 27 los predicados, en la figura 28 las acciones y en la figura 29 las relaciones entre los conceptos de la ontología definida para el sistema.

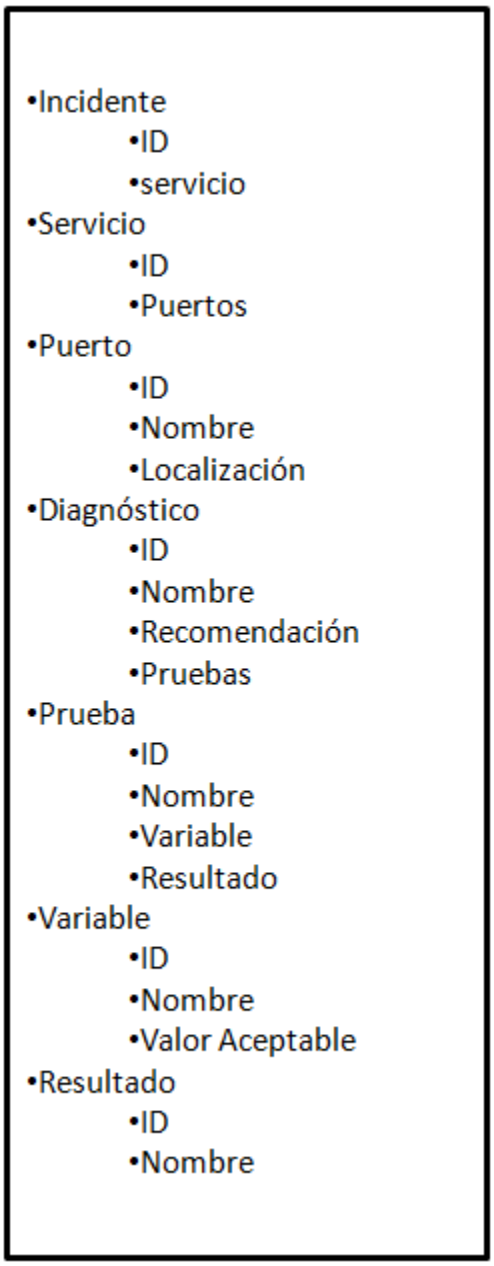

Figura 26. Conceptos en la ontología.

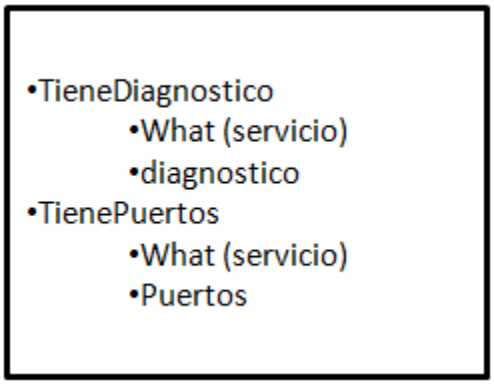

Figura 27. Predicados de la ontología.

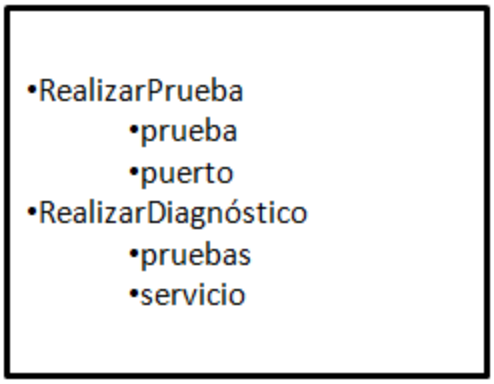

Figura 28. Acciones de los agentes.

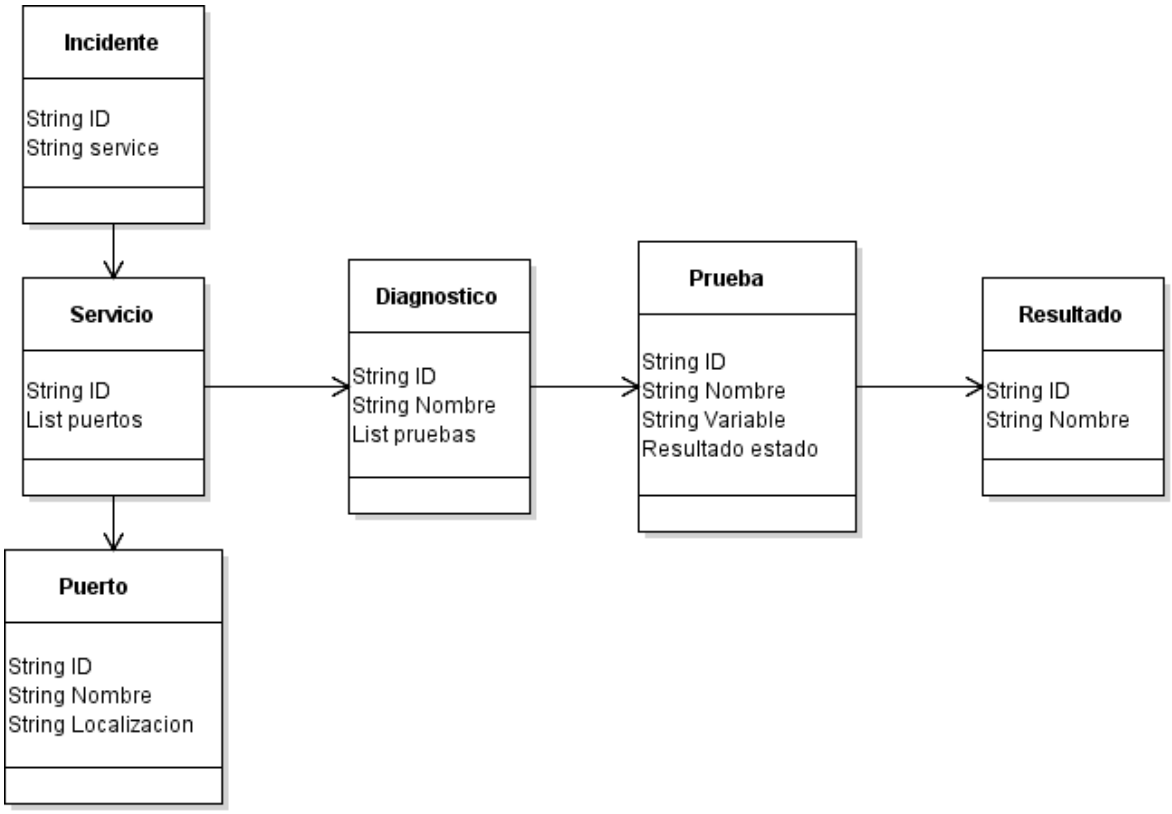

Figura 29. Ontología de diagnóstico de incidentes.

5.5.10 Selección del lenguaje de contenido

El décimo paso, según Nikraz, M. et al. [66], consiste en seleccionar el lenguaje a utilizar en los mensajes de las interacciones entre los agentes. JADE provee dos lenguajes de contenido: el SL y el LEAP. El lenguaje SL es legible, mientras que el LEAP está codificado en bytes. Las reglas de decisión para la elección de un lenguaje que se utilizan son: el SL es apropiado para aplicaciones abiertas, es decir que requieren que agentes de diferentes desarrolladores que corran en diferentes plataformas se puedan comunicar. LEAP es más liviano, por lo que se recomienda en ambientes donde se desee minimizar el uso de memoria.

En el presente trabajo, se escogió el lenguaje SL, debido a que se utiliza en ambientes que no incluyan equipos móviles y se espera legibilidad del lenguaje con fines de seguimiento y depuración.

En este capítulo se siguieron los pasos de análisis y diseño de la metodología propuesta por Nikraz el al. para el desarrollo de MAS y se llegó a la definición de responsabilidades, interacciones, estructura de mensajes de los agentes a usar en el sistema. En el próximo capítulo, se presenta cómo a partir del análisis y el diseño, se construyó el sistema.

6. Implementación

En este capítulo se presenta el proceso de implementación del sistema multiagente. Se mencionan las herramientas de programación, librerías y principios usados en este proceso. Para describir la solución implementada, se incluyen los gráficos indicados por Krunchten [68] en su arquitectura “4+1”, a saber, la vista lógica, física, de proceso, de despliegue y de casos de uso.

6.1 Herramientas usadas

La implementación se realizó utilizando la herramienta JADE para proveer los contenedores, los agentes de páginas amarillas y la creación y comunicación entre agentes. El lenguaje utilizado fue Java y el entorno integrado de desarrollo, Eclipse [69].

Para que el MAS se comunique con la aplicación web, se crearon servicios web usando el módulo adicional WSIG (*Web Service Integration Gateway*) de JADE. Para almacenar en una base de datos el estado de los agentes y los resultados de sus acciones, se utilizó el módulo adicional *Persistence* de JADE.

La aplicación web que sirve de interfaz para el usuario, se desarrolló utilizando la librería Vaadin. Esta librería permite crear aplicaciones con interfaz web con código en Java.

6.2 Descripción de la solución

La solución a nivel lógico se representa como un conjunto de clases. Se incluyen jerarquías correspondientes a los agentes, que descienden de la clase Agent de JADE. Cada clase de agente implementa las responsabilidades determinadas en el diseño, el *device*, interactúa con los dispositivos y responde a peticiones de ejecución de comandos, *callcenter* para responder a las peticiones de diagnóstico del usuario, *diagnosticator* coordina peticiones a otros agentes para entregar el diagnóstico del servicio. Para la ejecución de comandos en los dispositivos la clase Probe (prueba) usa las clases de la jerarquía Cmd (comandos). La clase *diagnosticador* utiliza el motor de reglas

(RuleEngine) y la jerarquía descendiente de DiagRule (regla de diagnóstico). Las clases usadas en el proyecto se presentan en la figura 30.

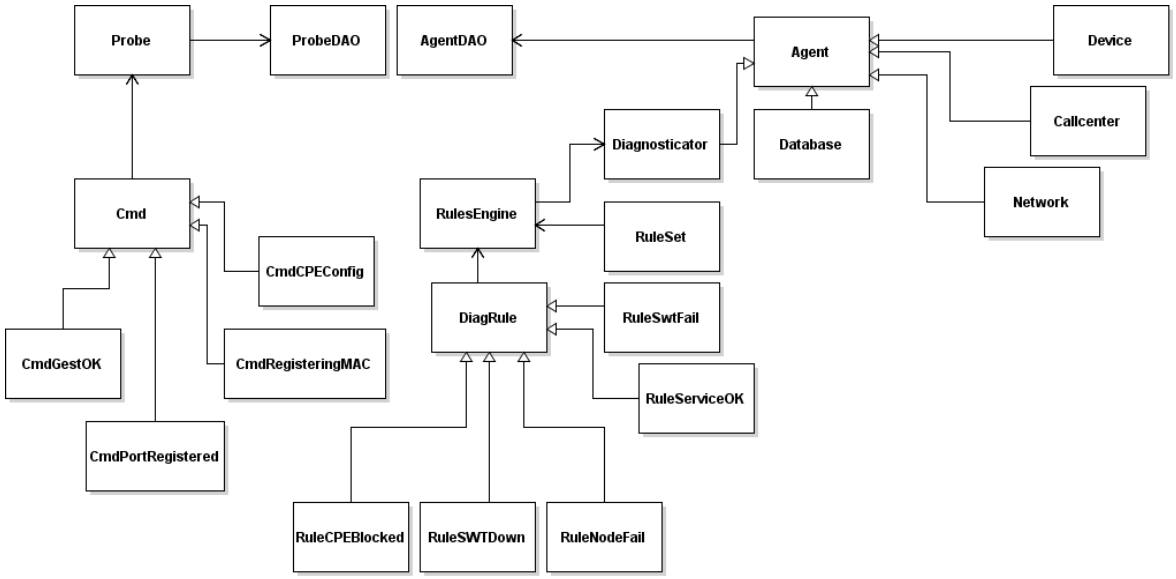

Figura 30. Clases del sistema.

Desde el punto de vista de procesos, el sistema debe proveer un diagnóstico del estado de un servicio en particular; para ello, debe recibir peticiones de diagnóstico, ejecutar pruebas sobre los dispositivos, analizar los resultados de estas pruebas, usar reglas de decisión para llegar a un diagnóstico y luego entregarlo al usuario. Esto se representa gráficamente en la figura 31.

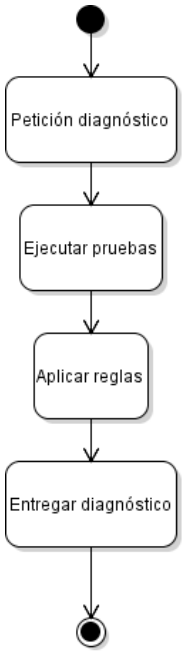

Figura 31. Proceso de diagnóstico.

Desde el punto de vista de desarrollo, el sistema se implementó usando los paquetes de agentes, servicios web, GUI (*Graphic User Interface*), el de pruebas sobre dispositivos (probes), comandos, reglas y bases de datos. El paquete Agents contiene los agentes *callcenter* (encargado de disparar el proceso de diagnóstico), *device* (encargado de la conexión a dispositivos), *network* (encargado de la creación de los dispositivos) y *database* (encargado de la conexión a bases de datos). El paquete *Commands* contiene la lógica para ejecutar pruebas. El paquete *Diagnostic* usa el resultado de las pruebas y un conjunto de reglas para entregar el diagnóstico. La representación gráfica de estos

paquetes, se puede apreciar en la Figura 32.

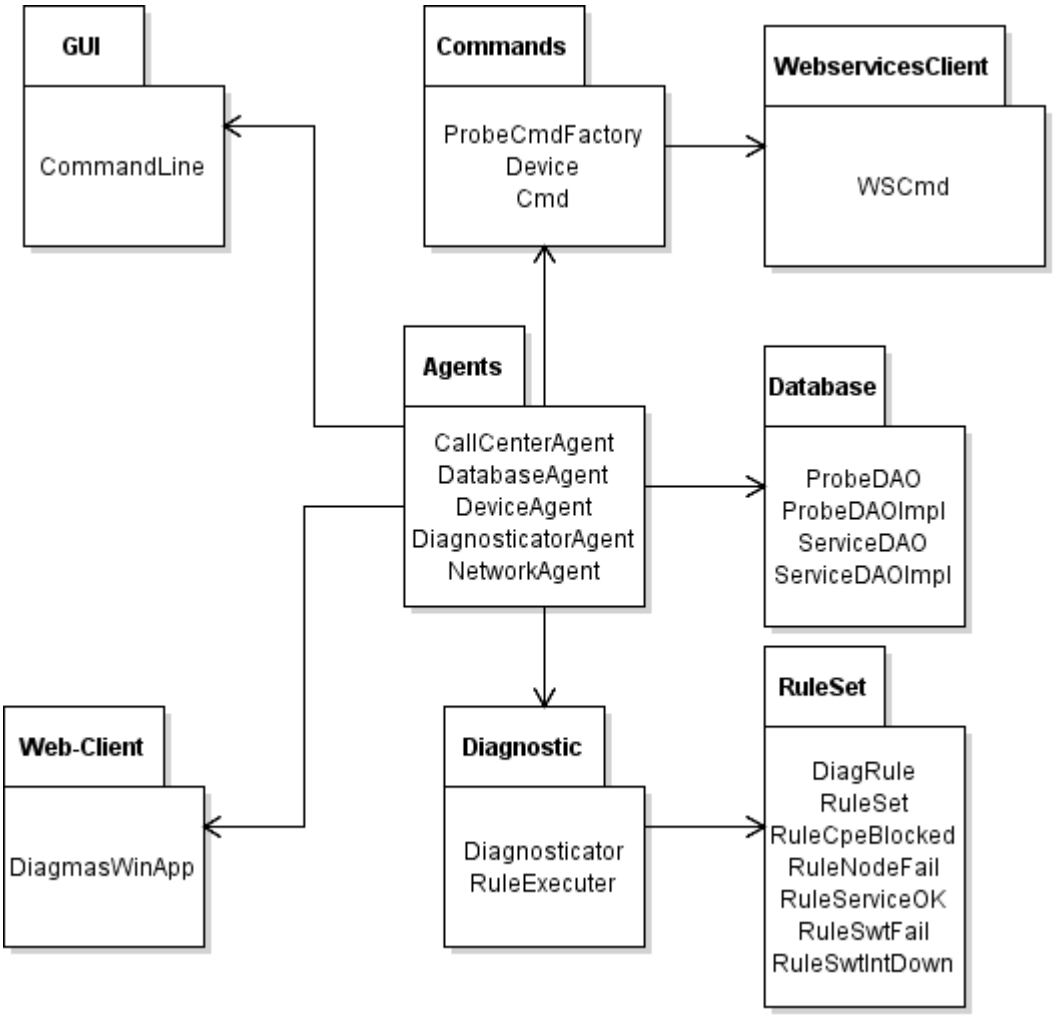

Figura 32. Paquetes del sistema.

Desde el punto de vista físico, el sistema utiliza dos servidores, uno para la aplicación web (GUI) y otro para el MAS. El MAS se encarga de hacer peticiones a cada agente de dispositivo, para determinar el estado de los servicios prestados. Los dispositivos son accedidos por medio de una conexión a la red de la compañía. La información de los servicios es obtenida de la CMDB. Los usuarios utilizan clientes web para interactuar con el sistema. Esta estructura se puede apreciar en la figura 33.

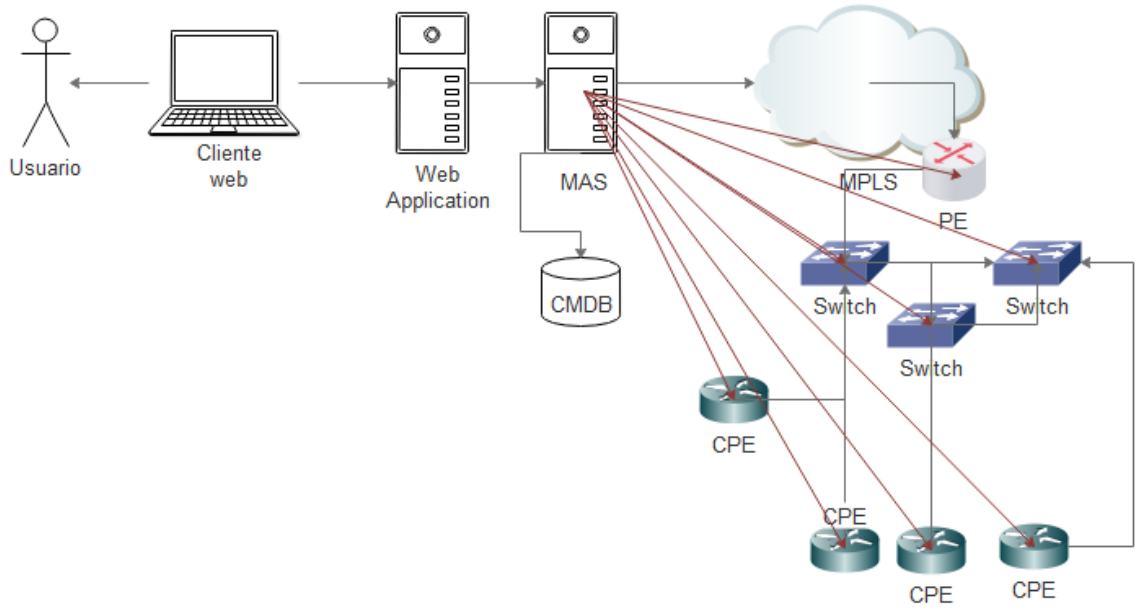

Figura 33. Hardware usado en el sistema.

Desde el punto de vista de los escenarios, se encuentran el diagnóstico y sus casos de uso incluidos. Estos son ejecución y análisis de las pruebas, selección del diagnóstico y registro histórico del mismo. El coordinador genera reportes de uso, por medio de la consulta de históricos. Estos escenarios se pueden observar como casos de uso en la figura 34.

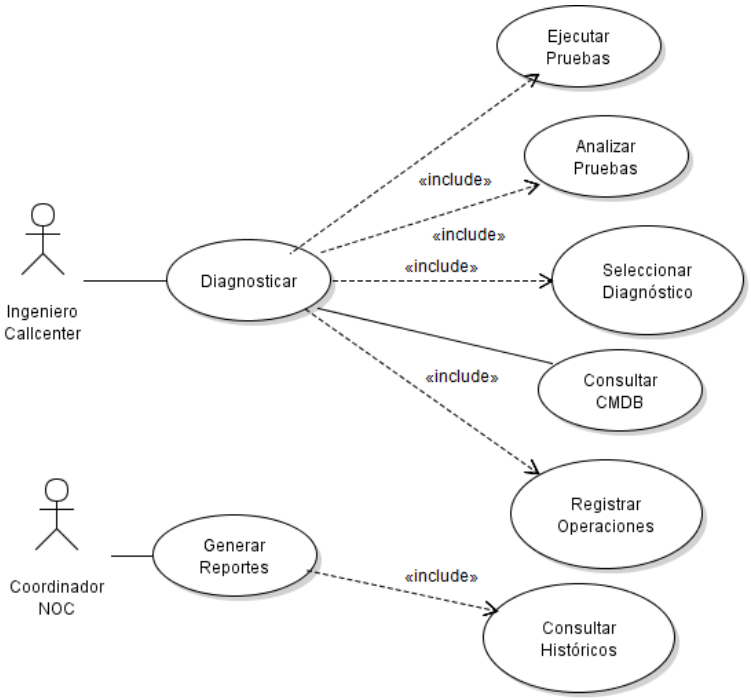

Figura 34. Casos de uso del sistema.

6.3 Modelado de las soluciones brindadas a los incidentes

Según Knaeuper et al. [70], el modelo de representación de conocimiento más comúnmente implementado es el sistema de producción. Ese consiste básicamente de dos partes: 1) Un conjunto de reglas de la forma IF(situación), THEN (acción) o (situación) y 2) Una estructura de control para administrar las reglas.

En el presente trabajo se escogió como modelo para modelar las soluciones brindadas a los incidentes como un sistema de producción. De esta forma las reglas se modelan usando el motor de reglas de java *Easyrules* [71]. Según su documentación, *EasyRules* provee una abstracción de la Regla para crear reglas con condiciones y acciones, y adicionalmente el motor de reglas, *RulesEngine*, que corre a través de un conjunto de reglas para evaluar las condiciones y ejecutar sus acciones. Una representación de alto nivel de este modelo se puede observar en la Figura 35.

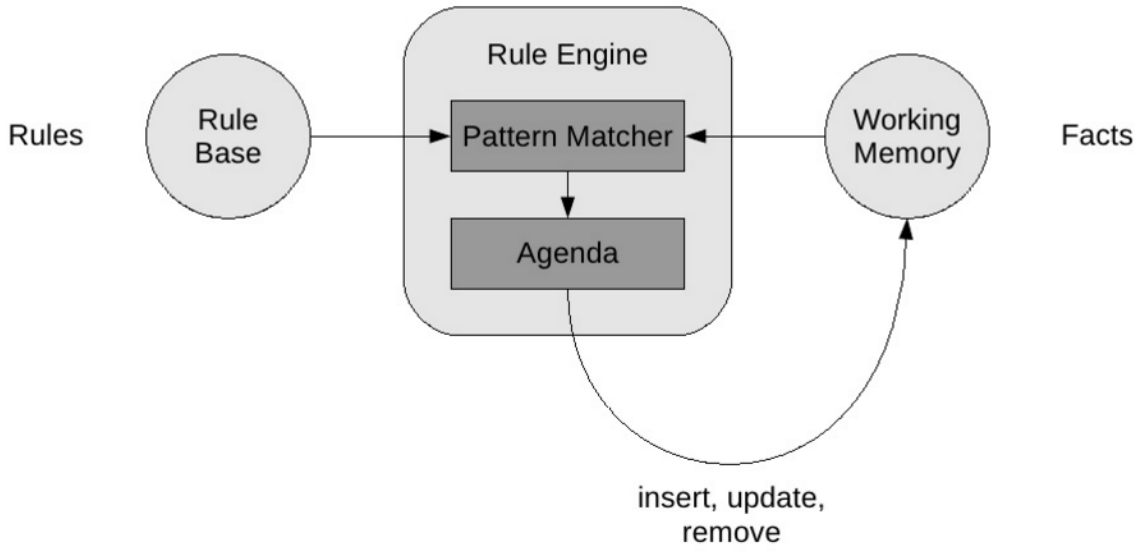

Figura 35. Representación de un sistema de producción de reglas. Tomado de Drools Team, **[72]**.

6.4 Software desarrollado

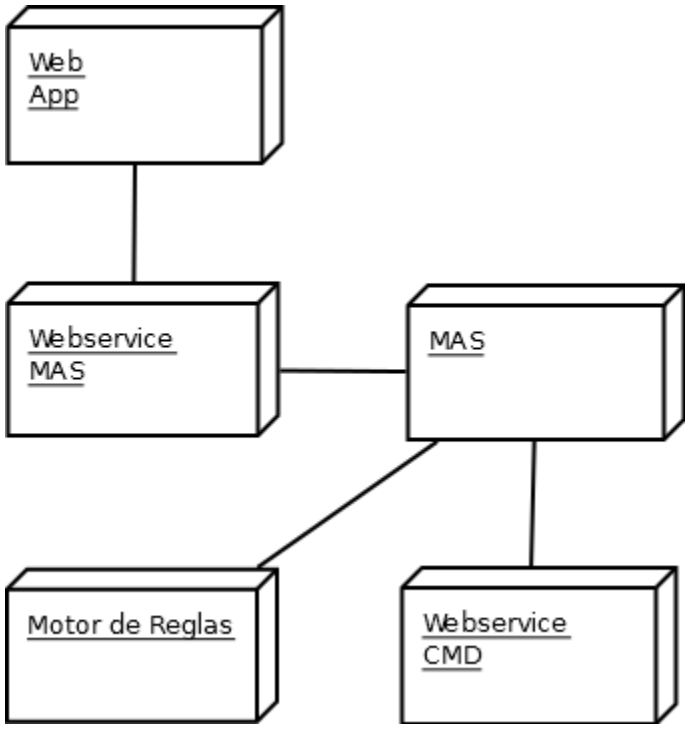

Figura 36. Diagrama del software desarrollado. (Creado por el autor).

En la figura Figura 36. Diagrama del software desarrollado. (Creado por el autor).se representa el software que se desarrolló para el proyecto. los bloques son:

Web App: Aplicación Web. Se encarga de recibir las peticiones de diagnóstico, llamar el webservice que utiliza el MAS y presenta las respuestas del mismo al usuario.

Webservice MAS: Servicio web que se conecta al MAS y envia mensajes a los agentes para que inicien su labor.

MAS: Sistema Multi-agente. Se encarga de realizar las pruebas a los dispositivos y de realizar el diagnóstico.

Motor de Reglas: Conjunto de reglas a ser aplicadas para llegar a un diagnóstico.

Webservice CMD: Servicio web que permite ejecutar comandos en dispositivos remotos. El servicio web recibe como argumentos la ip del dispositivo, credenciales y comando a ejecutar. Retorna el resultado de la ejecución del comando. El servicio web se desarrolla

para un contenedor tomcat. Para realizar la ejecución, se utiliza a su vez, un programa en python: runcmd.py. El script se incluye como Anexo 1.

6.5 Implementación del sistema usando principios de una metodología ágil

Para la implementación del MAS se utilizaron principios de Scrum [73]. Scrum es un marco de trabajo en el cual las personas pueden enfrentar problemas complejos, mientras productiva y creativamente entregan productos del más alto valor posible. Scrum emplea un enfoque iterativo e incremental para aumentar la predictibilidad del desarrollo y controlar los riesgos.
En Scrum, el objetivo global del proyecto se divide en objetivos de menor tamaño que son adicionados a la lista de requerimientos a ser implementados. Esto se realiza en iteraciones (*sprints*) [73].

Los tres principios de Scrum que se utilizaron en el proyecto fueron: transparencia, inspección y adaptación.

La transparencia se aplicó en que tanto para el usuario como para el desarrollador, se compartía el objetivo del desarrollo así como los términos utilizados en la implementación. Los ingenieros y el coordinador del NOC, tuvieron acceso a los entregables del desarrollo desde el primer *sprint*.

La inspección consistió en que los ingenieros y el coordinador NOC frecuentemente fueron informados sobre el avance del desarrollo y éstos pudieron revisar los resultados parciales del proceso.

La adaptación se refiere a que el producto fue ajustado, debido a que en algún momento se desviaba de los criterios de los usuarios. Estos ajustes fueron hechos lo más temprano posible para minimizar los efectos adversos sobre el proyecto.

Para aplicar los principios de inspección y adaptación, se realizaron a nivel de *sprint* las tareas de planeación de *sprint*, revisiones y retrospectivas, así como scrum semanal (*weekly scrum)* en lugar de hacerlo a diario.

El propietario del producto fué el Coordinador del NOC. Éste fué el responsable de validar la entrega. Es decir, responder a la pregunta: ¿se construyó el producto correcto?

El desarrollo se realizó en cinco sprints, con una duración de aproximadamente un mes cada uno, en el primero se implementaron los agentes Device y su comportamiento. En el segundo, los comandos a ejecutar en los dispositivos. En el tercero el conjunto de reglas. En el cuarto los servicios web. En el quinto la aplicación web, que sirve de GUI y se realizó la integración.

6.6 Uso de la herramienta JADE

JADE [74] es un framework implementado en el lenguaje Java. JADE simplifica la implementación de MAS a través de un software que se encuentra entre el sistema operativo y las aplicaciones (*middleware)* que cumple con las especificaciones de la FIPA y a través de un conjunto de herramientas gráficas que soportan tareas de desarrollo y depuración.

Además de la abstracción de agentes, JADE provee un modelo de ejecución de tareas simple pero poderoso, un mecanismo de comunicación entre agentes basado en el envío de mensajes asíncronos, un servicio de páginas amarillas que soporta el mecanismos de descubrimiento y otras funcionalidades como facilidades para la creación de servicios web y proveer persistencia de los agentes a través del módulo adicional *Persistence* [74].

En la figura 37, se puede observar la interfaz de JADE. En ella se encuentran las herramientas para crear, ejecutar, enviar mensajes y disponer de los agentes.

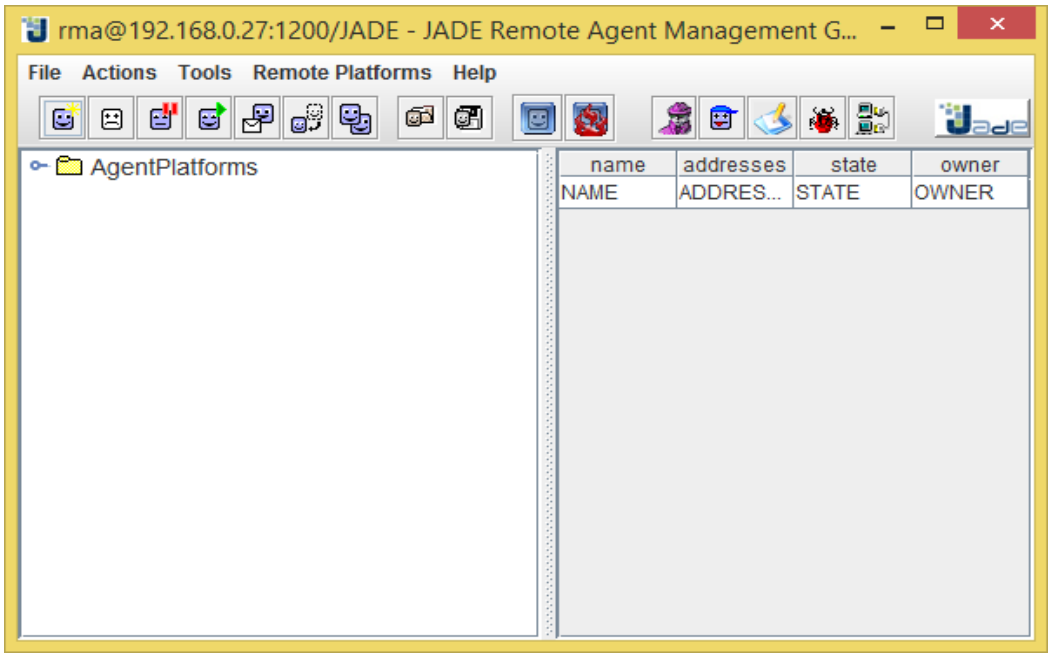

Figura 37. Interfaz de JADE.

En la figura 38, se puede observar un contenedor con los agentes tipo Device que se conectan a los dispositivos de la red. Adicionalmente, el agente callcenterJOE es un agente de callcenter, y el agente WIN0003 es un agente diagnosticador.

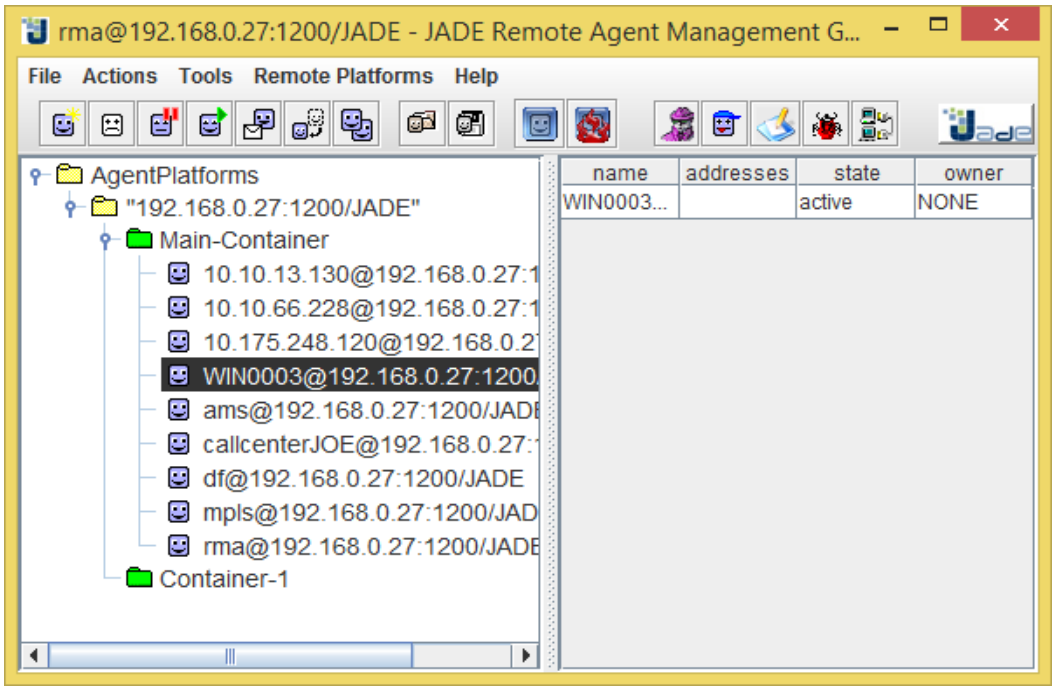

Figura 38. Agentes en el contenedor.

En la figura 39, se puede observar un diálogo que permite enviar mensajes a agentes. Esta funcionalidad se reproduce en el código implementado.

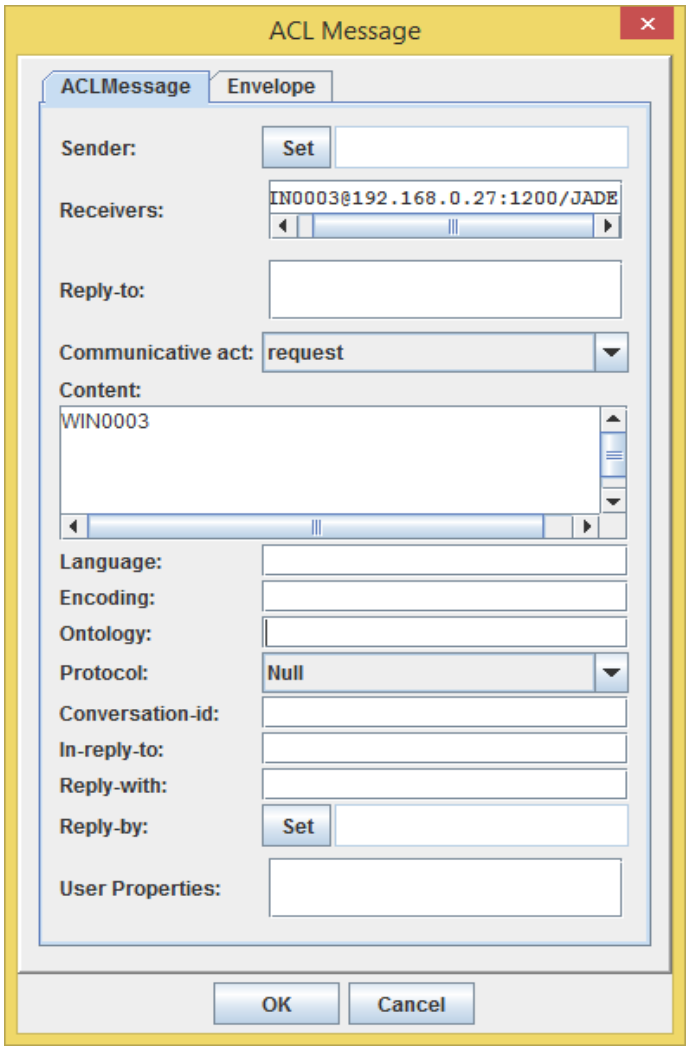

Figura 39. Envío de mensajes en JADE.

En la figura 40, se puede observar una pantalla de eclipse, un entorno integrado de desarrollo, en el que aparece la estructura de paquetes y algunas clases codificadas del proyecto.

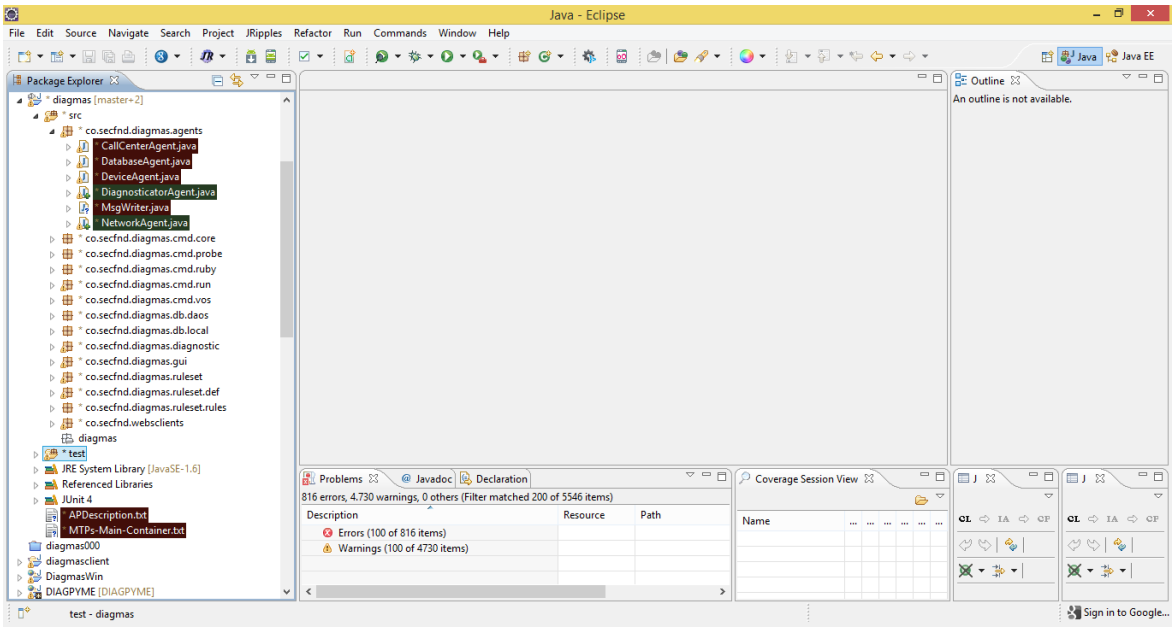

Figura 40. Proyecto en eclipse.

Dos pantallas de la aplicación web desarrollada para que permita la interacción con los ingenieros *callcenter*, la pantalla de conexión al sistema y la pantalla de inicio de diagnóstico pueden apreciarse en la figura 41.

Figura 41. Conexión de la aplicación web.

En este capítulo se presentaron detalles del proceso de implementación, los principios seguidos y las herramientas utilizadas para el mismo. En el siguiente capítulo, se presentan las pruebas realizadas al sistema.

7. Pruebas

En este capítulo se presentan los resultados de las pruebas del sistema. Se describen los escenarios y las pruebas realizadas. Como resultado de las pruebas, el sistema funcionó correctamente, proporcionando diagnósticos acertados en los escenarios definidos.

7.1 Definición de pruebas

Las pruebas realizadas se clasifican como unitarias, de integración y del sistema. Estas pruebas fueron creadas usando JUnit 4 [75] y se buscó una cobertura de más del 80% del código desarrollado con las pruebas unitarias. Para las pruebas del sistema se usó una red simulada así como un conjunto de escenarios predefinidos para los cuales el diagnóstico era conocido. Para realizar las pruebas se invoca el servicio web de diagnóstico por medio de la herramienta SoapUI [76] y se observa la respuesta.

7.2 Definición de red de dispositivos de pruebas

Para realizar las pruebas del sistema, se crearon escenarios simulados con las condiciones a ser diagnosticadas. Para ello se utilizó el software GNS3 [77], la cual permite construir, diseñar y probar configuraciones de red. En esta herramienta se crea una red de dispositivos de pruebas como un laboratorio en GNS3. Esto se puede observar en la figura 41. Este laboratorio contiene un subconjunto de la red MPLS con ocho equipos: un *router* PE, cuatro *switch* y tres CPE. Cada dispositivo virtual reproduce los comandos del dispositivo real, pues se le carga una copia del *firmware* utilizado por el equipo real. Así mismo, simula la adición de módulos y la configuración en tiempo de ejecución y en memoria NVRAM.

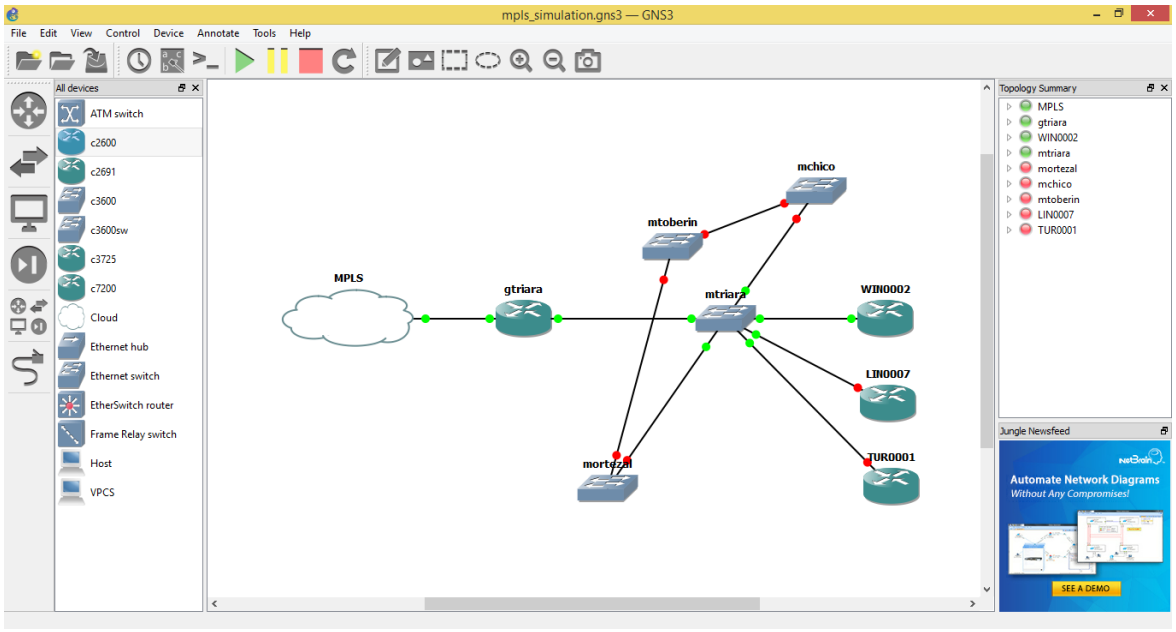

Figura 42. Red de pruebas.

Los dispositivos a formar parte de la prueba se pueden observar en la figura 42. Se utiliza un *router* Cisco 12410, que funciona como *router* de borde. Un *switch* Catalyst C4510 para el nivel de acceso y *routers* cisco 2921 para los equipos del lado del cliente.

Dispositivo	Marca	Imagen	Modelo	Imagen IOS
Router PE	Cisco		3725	c3725-adventerprisek9-mz124-15.image
Switch	Cisco		3640	c3640-ik9o3s-mz124-13.image
CPE	Cisco		2691	c2691-entservicesk9-mz.124-13b.image

Figura 43. Equipos a utilizar en las pruebas.

7.3 Escenarios de prueba

Con el fin de probar cada diagnóstico, se crearon tantos escenarios como diagnósticos tiene el sistema. Estos escenarios reproducen las condiciones necesarias para cada diagnóstico. Cada escenario se crea como un archivo para GNS3 con las características descritas en la figura 44. En todos los escenarios de pruebas el sistema respondió con el diagnóstico correcto, según los ingenieros NOC consultados.

Diagnóstico	Archivo	Descripción
Ultima milla OK	SERVICE_OK.gns3	Nodo arriba, Switch arriba, CPE arriba, equipos detrás del CPE arriba
Bloqueo	CPE_BLOCKED.gns3	Ataque DOS a CPE
Error Configuración	CPE_CONFIG_ERR.gns3	Configuración errónea en direccionamiento equipos detrás del CPE.
Falla Fibra	FIBER_FAIL.gns3	Anillo de switch abierto
Falla Nodo	NODE_FAIL.gns3	Interfaz nodo apagada
Tabla ARP vacia	SWT_ARP_FAIL.gns3	Equipos detrás del CPE apagados
Interfaz caida	SWT_INT_DOWN.gns3	Interfaz switch apagada
Falla Switch	SWT_FAIL.gns3	Switch apagado

Figura 44. Escenarios de prueba.

En la figura 45 se puede observar un ejemplo de la configuración de un dispositivo en el simulador GNS3.

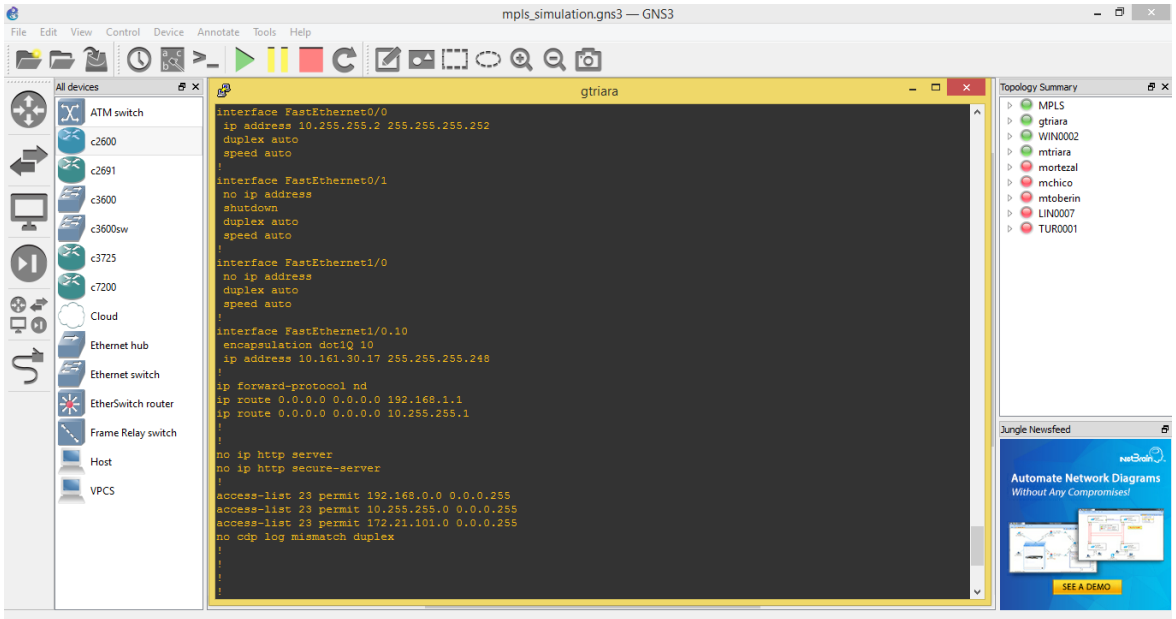

Figura 45. Parte de la configuración de un dispositivo de pruebas.

7.3.1 Escenario de servicio sin falla.

Para que un servicio se encuentre sin falla, se requiere que los equipos *router* PE, *switch* y CPE se encuentren encendidos, que el puerto esté registrado en el *switch*, que el puerto del *switch* tenga en su tabla de MAC las direcciones de dispositivos del lado del cliente y que los comandos con la herramienta ping hacia la dirección IP de la interfaz WAN del CPE respondan sin pérdida de paquetes. En la figura 46 se observan los resultados de la consulta del servicio web para este escenario.

El sistema responde con el diagnóstico SERVICE_OK, el cual indica que el servicio está activo y que no se debe realizar ninguna acción correctiva. En la figura 47 se puede observar el estado de la interfaz en el switch.

Figura 46. Resultado para el escenario sin falla.

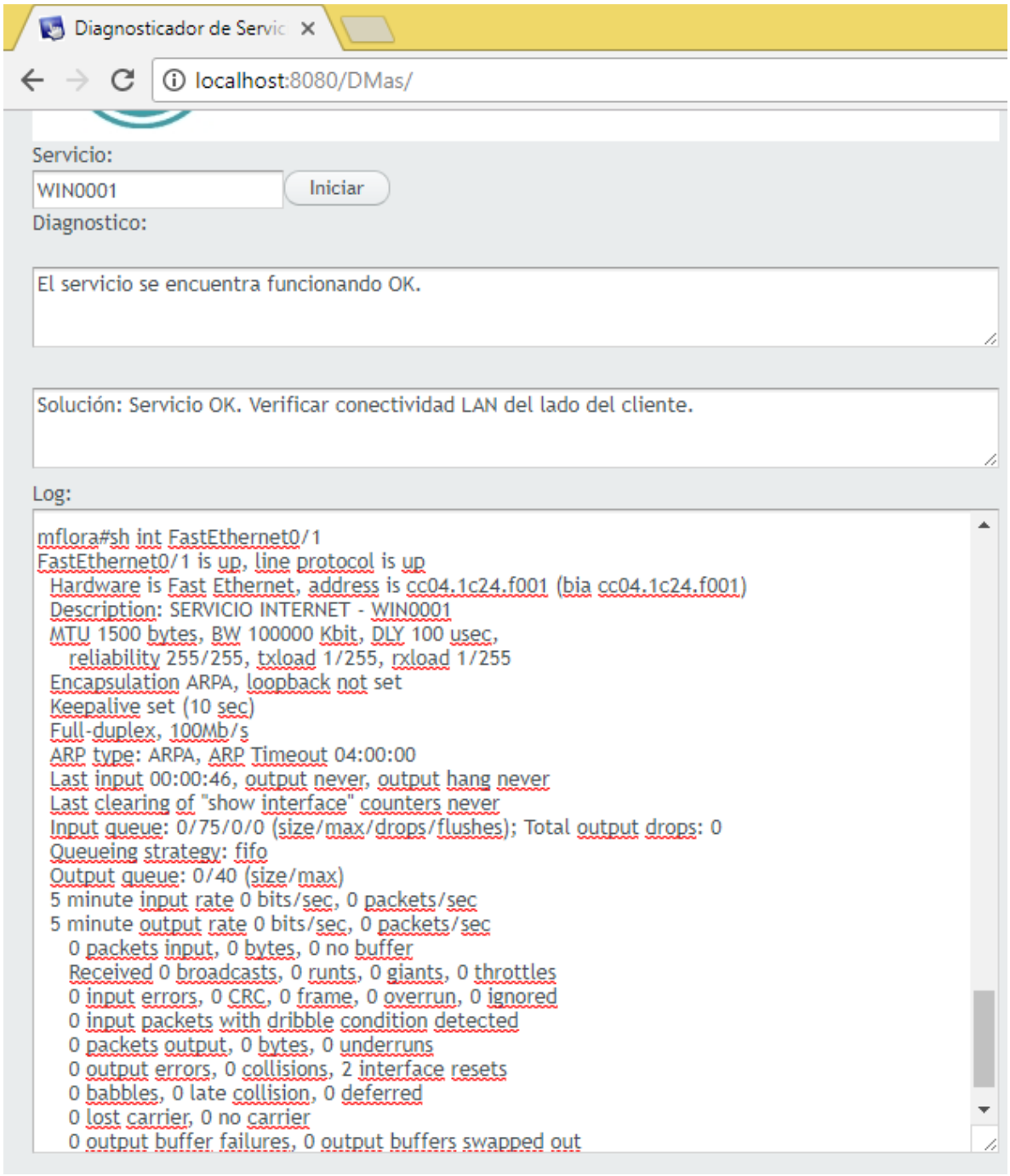

Figura 47. Resultado de la prueba en el switch.

7.3.2 Escenario de CPE bloqueado

Para representar la condición de un CPE bloqueado se debe cumplir que en el puerto de *switch* aparezcan registradas direcciones MAC de los equipos del lado del cliente, pero que cuando se envíen paquetes de prueba usando la herramienta *ping*, éstos se pierdan y no sea respondida la consulta. Para simular esto, el CPE debe ejecutar un script que consuma sus recursos, el script contiene los comandos **show tech-support, write memory, show-running configuration y debug all.** Los resultados de consumir el servicio web se pueden ver en la figura 48. En este escenario la respuesta del sistema es CPE_BLOCKED, indicando que el CPE debe ser reiniciado.

Figura 48. Respuesta para el escenario de CPE bloqueado.

7.3.3 Escenario de CPE con error en su configuración

Para representar la condición de un CPE con error en su configuración, se deshabilita su enrutamiento. El *router* PE debe estar configurado correctamente así como el *switch*. El resultado del consumo del servicio web se puede ver en la figura 49. El sistema responde con el diagnóstico CPE_CONFIG_ERROR, lo cual indica que el ingeniero NOC debe revisar la configuración del CPE para que la falla pueda ser solucionada. En este escenario, el sistema proporciona el diagnóstico apropiado.

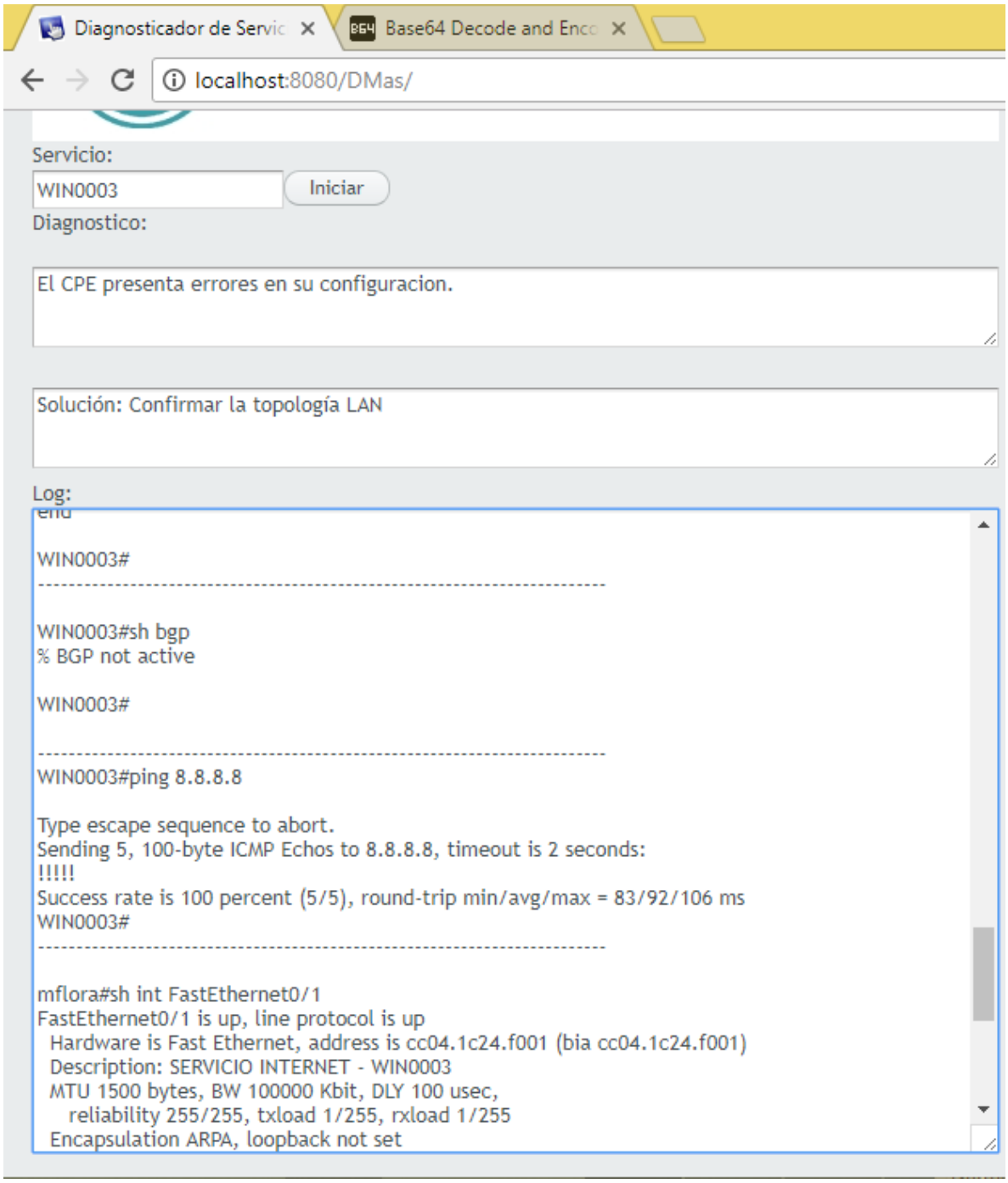

Figura 49. Respuesta para el escenario de CPE con error en su configuración.

7.3.4 Escenario de falla por corte de fibra

Para representar el corte de fibra, se elimina la conexión entre la interfaz WAN del CPE y el *switch*. El *router* PE y el *switch* deben estar configurados como en el escenario sin falla. El resultado del consumo del servicio web en la figura 50.

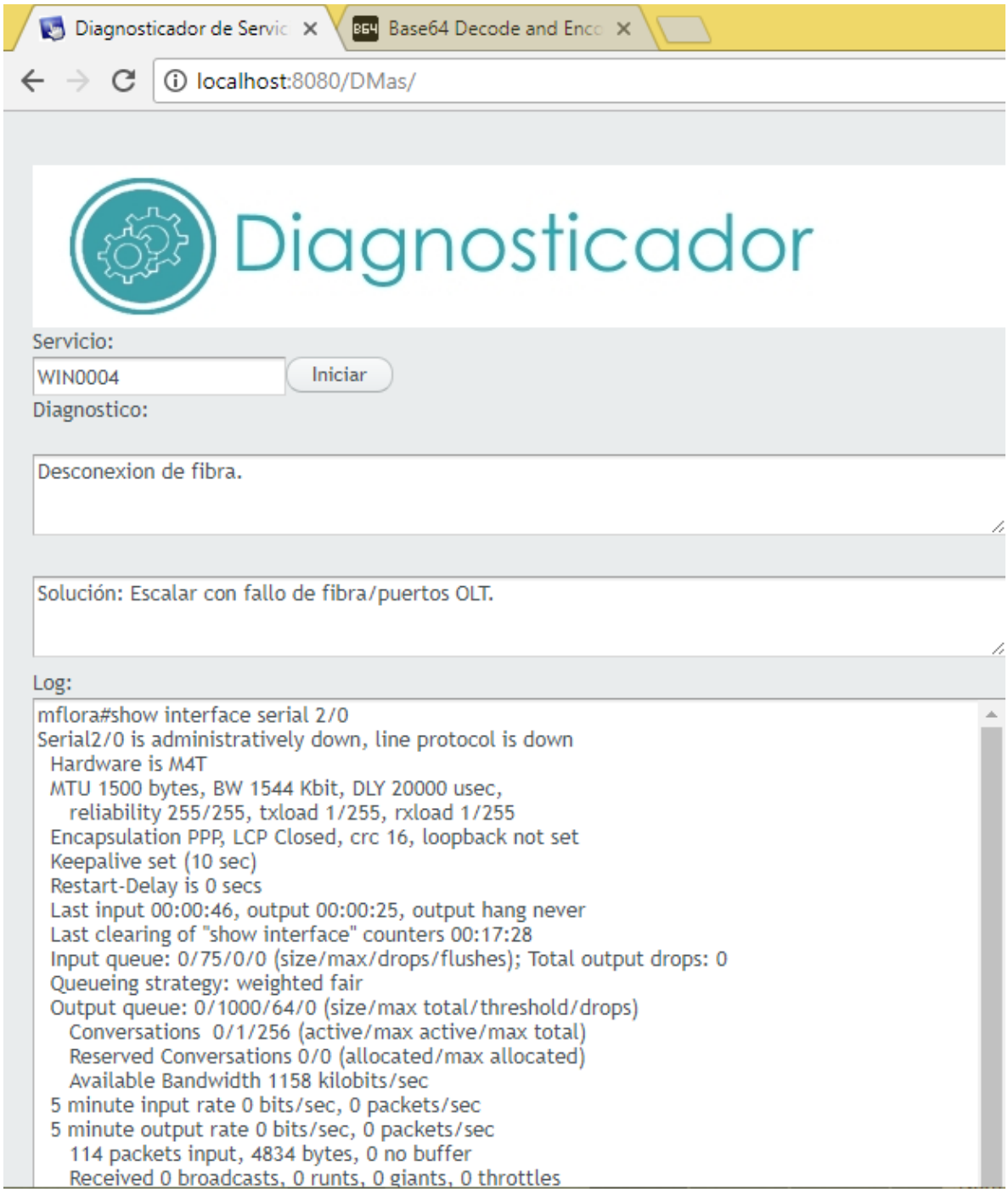

Figura 50. Respuesta para el escenario falla por corte de fibra.

7.3.5 Escenario de falla en *router* PE

Para representar la falla en el *router* PE, el equipo se apagará. El resto de los equipos se dejarán funcionando correctamente. El resultado del consumo del servicio web se observa en la figura 51. El sistema responde indicando que el PE está fallando, lo cual alertará al ingeniero NOC para que lo revise. El sistema respondió correctamente.

Figura 51. Respuesta para el escenario de falla en *router* PE.

7.3.6 Escenario de falla por tabla ARP sin registrar dispositivos en *switch*

Para representar la falla en el *switch*, la interfaz se configura sin VLAN, mientras que el CPE y el PE deben estar encendidos y configurados correctamente. El resultado del consumo del servicio web se aprecia en la figura 52. En este caso se observa la respuesta SWT_ARP_FAIL, que indica que la tabla de MAC de la interfaz del *switch* no está registrando dispositivos del lado del cliente, por lo que deben realizarse acciones correctivas a nivel del *switch*.

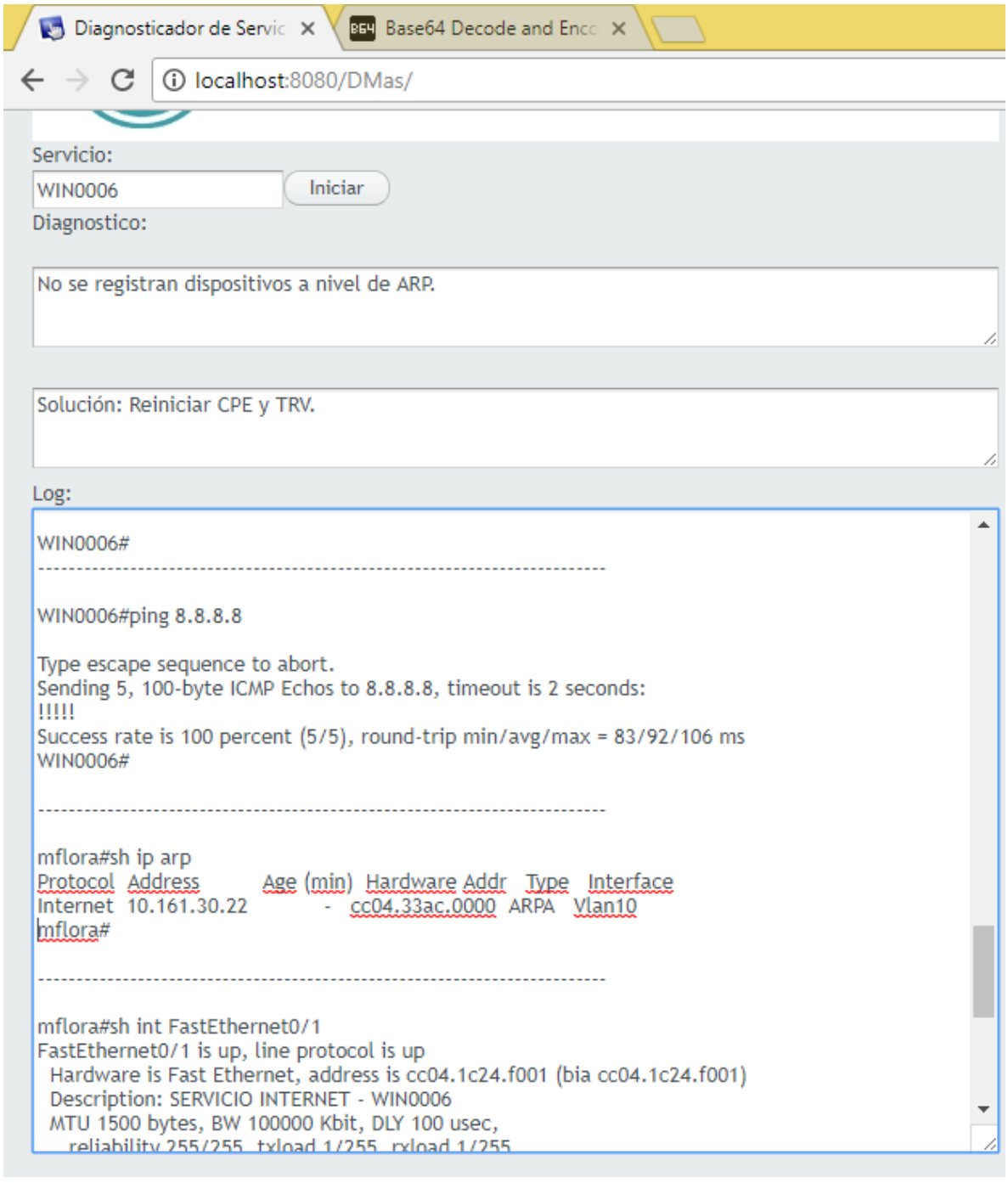

Figura 52. Respuesta escenario sin dispositivos a nivel ARP en *switch*.

7.3.7 Escenario de falla por interfaz *switch* apagada.

Para representar la falla en el *switch*, la interfaz se coloca operativamente down. El router PE y el CPE se configuran correctamente. El resultado del consumo del servicio web se puede apreciar en la figura 53. Aquí el sistema responde con SWT_INT_DOWN lo que indica que acciones correctivas deben ser tomadas en la interfaz del *switch*. Además, indica que el sistema está respondiendo de la manera esperada.

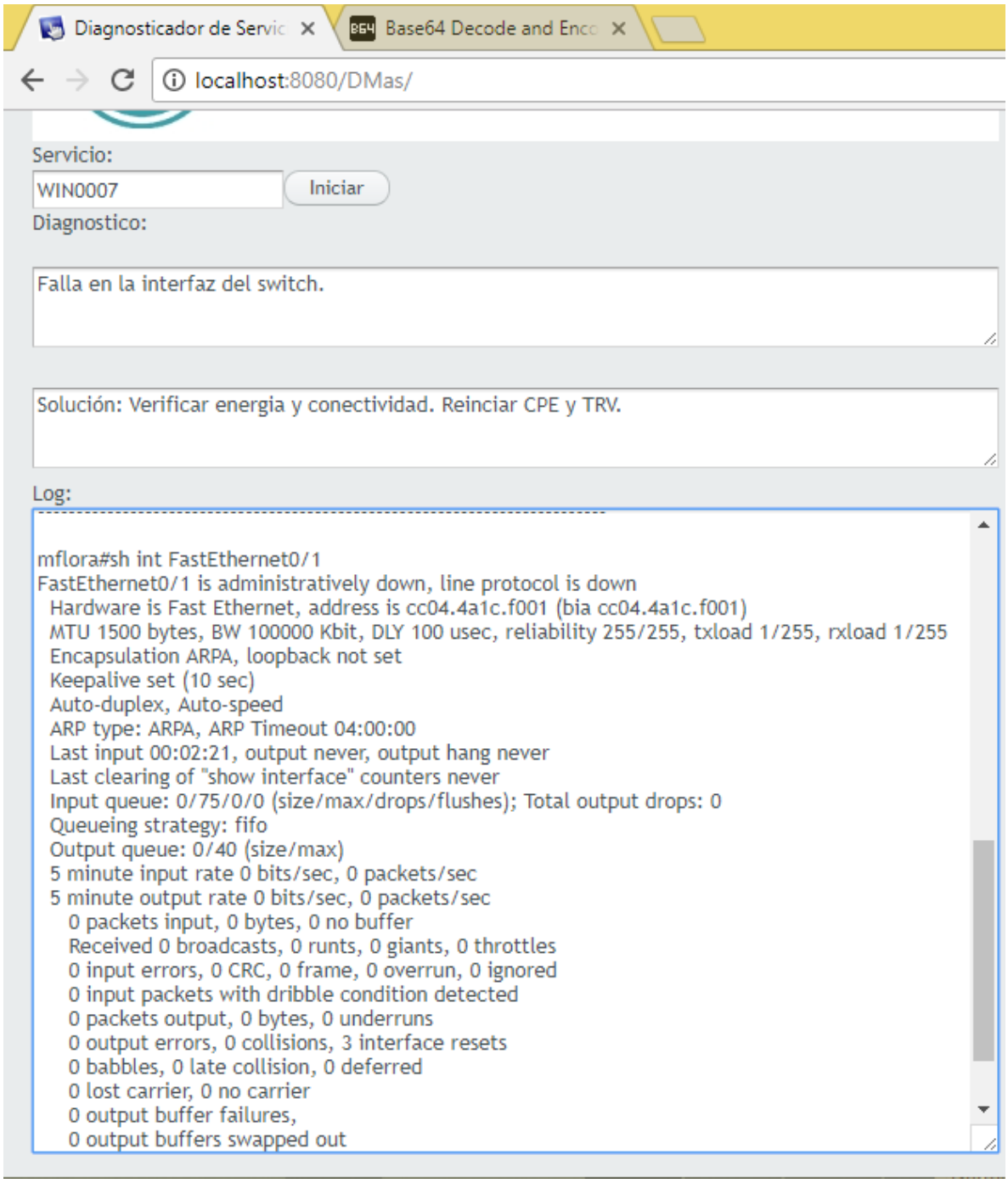

Figura 53. Respuesta para escenario de interfaz de switch apagada.

7.3.8 Escenario de falla por *switch* apagado

Para representar la falla en el *switch*, éste se apaga. El router PE y el CPE se configuran correctamente. La respuesta del consumo del servicio web se puede observar en la figura 54. El sistema responde con SWT_FAIL, lo cual indica que no se pudo conectar al *switch* o que éste está apagado, por ello los esfuerzos del ingeniero NOC deben dirigirse a ese dispositivo. La respuesta es la esperada por lo que el sistema funciona correctamente.

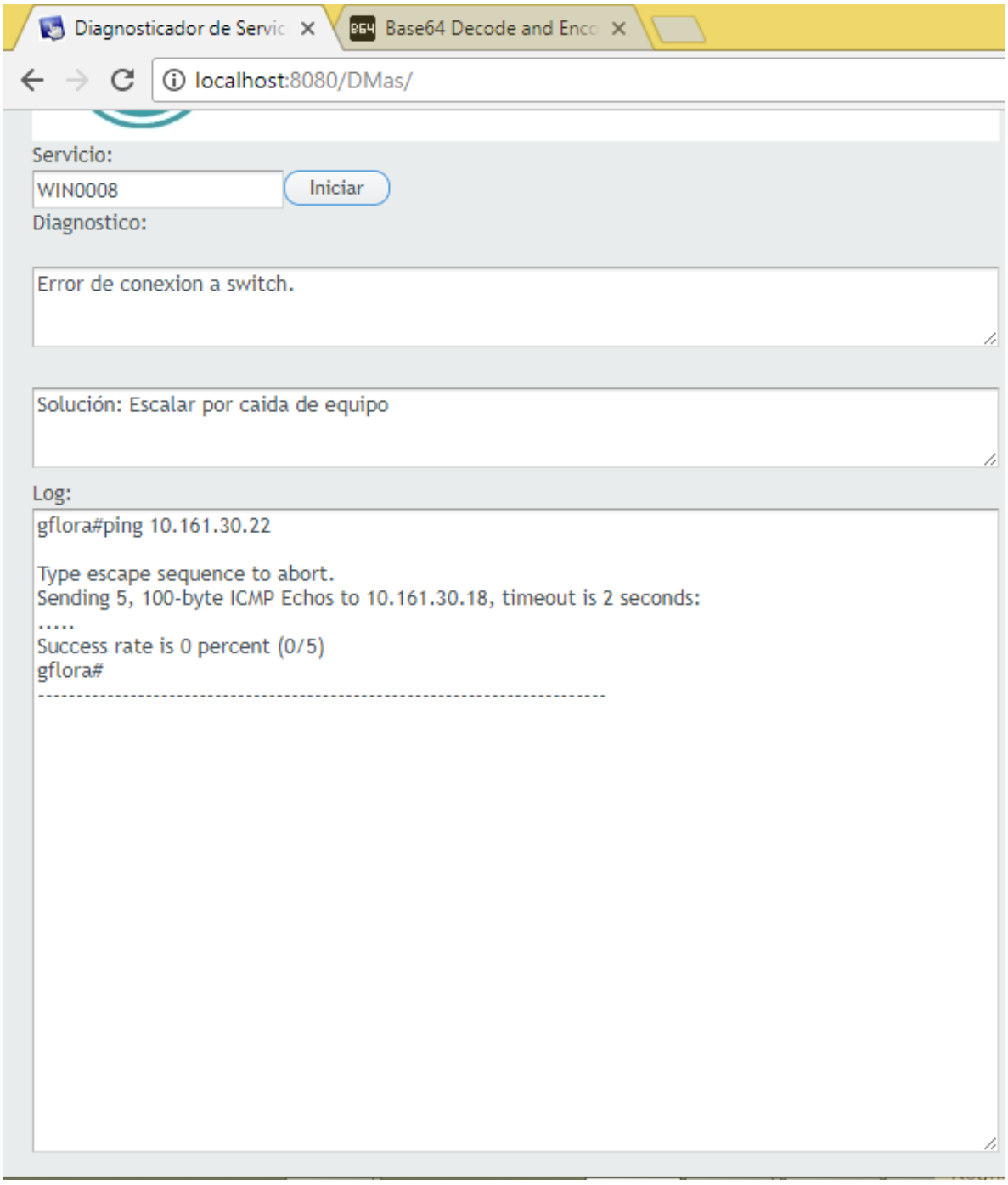

Figura 54. Respuesta para escenario de switch apagado.

En este capítulo se presentaron las pruebas realizadas al sistema, en ellas se evidencia que el sistema realiza los diagnósticos en los escenarios predefinidos, cumpliendo así con

los requerimientos. En el siguiente y último capítulo se presentan las conclusiones y recomendaciones producto del trabajo realizado.

8. Conclusiones y recomendaciones

En este capítulo se presentan las conclusiones que reúnen los resultados del trabajo, así como las lecciones aprendidas durante el desarrollo del mismo. Adicionalmente se incluyen recomendaciones para mejoras y trabajos futuros.

8.1 Conclusiones

La construcción de un sistema MAS implica la programación de agentes, para lo cual se hace necesario utilizar un marco de trabajo y la metodología orientada a agentes, en el presente trabajo el uso de la metodología de Nikraz et al y el marco de trabajo JADE, suplieron esta necesidad.

Un sistema MAS puede ser una alternativa no solo para el diagnóstico sino para el monitoreo y corrección automática de fallos en equipos de red siempre y cuando se desarrollen módulos de integración a los diferentes dispositivos así como el conjunto de reglas para disparar las acciones. Estos módulos deben permitir una conexión así como una extracción de la información necesaria para aplicar las reglas.

El uso de herramientas de simulación como Gsn3 permite realizar pruebas de sistemas con requerimientos de conexión a dispositivos Cisco, sin el sobrecosto de adquirir o utilizar la infraestructura real de una compañía con servicios de clientes para realizarlas.

Una alternativa de desarrollo de un sistema de diagnóstico sería adquirir y mantener una línea base del funcionamiento correcto de un servicio y una vez presentada la falla, realizar una comparación del estado actual contra dicha línea base. Sin embargo, la tarea de definición de los parámetros a medir, representaría el mayor reto debido a la heterogeneidad de las soluciones.

Un sistema MAS requiere una abstracción del mundo usando razonamientos locales simples en múltiples agentes. Esto facilita su aplicación en soluciones a problemas en

redes de computadores. Debido a su modularidad, las necesidades de escalamiento son satisfechas por la creación de más agentes.

Existe resistencia para la adopción de sistemas MAS. Esto debido a que, a pesar de sus ventajas, la recepción de una solución usando MAS en la empresa de telecomunicaciones fue vista con renuencia debido a que los proveedores establecidos en el mercado no las proporcionan y a que era considerada novedosa y poco probada.

Se requiere una mayor difusión y aplicación de MAS en las empresas de telecomunicaciones y en el ambiente corporativo. Esto debido a que en la empresa de telecomunicaciones, el sistema MAS fue considerado una novedad por la mayoría y aquellos quienes conocían la tecnología, le asociaban equivocadamente un gran esfuerzo y complejidad a la creación y mantenimiento de un MAS.

Se lograron determinar los incidentes más frecuentes y sus soluciones por medio de entrevistas con funcionarios de una empresa de telecomunicaciones, también se tuvo en cuenta la revisión histórica de incidentes.

Se modelaron las soluciones dadas a los incidentes más frecuentes por medio de un conjunto de reglas y un motor de evaluación de reglas. Estos toman los resultados de las pruebas ejecutadas en las máquinas y retornan un diagnóstico, el cual está asociado a una solución predeterminada, según lo establecido por los expertos humanos.

En el presente trabajo se analizó, diseñó, implementó y probó un sistema multi-agente para realizar diagnósticos de fallas de servicios de internet corporativo. En el proyecto se realizó un análisis descriptivo de los incidentes observados en una empresa de telecomunicaciones, gracias al mismo se pudieron determinar los incidentes más frecuentes y cuáles eran proclives a ser automatizados. Se utilizó la metodología de Nikraz et al. para el análisis y el diseño del proyecto. Esta metodología permitió llegar a un diseño de MAS para ser implementado usando la herramienta JADE. Para la implementación se utilizaron principios de transparencia, inspección y adaptación del

marco de trabajo Scrum. Esto permitió realizar la implementación con la participación del usuario final. Para la realización de pruebas se creó un ambiente simulado en GNS3 y escenarios con fallas configuradas para ser diagnosticadas por el sistema. El sistema pudo realizar los diagnósticos correctamente.

La dependencia de los expertos humanos, es decir, los ingenieros NOC y el coordinador del NOC, implica retos en cuanto a la disponibilidad de los mismos para responder a las preguntas que se les hacen, al establecimiento de un lenguaje común, a conocer las pruebas y los resultados que se esperan de las mismas, así como en la definición de reglas generales, pues existen diferencias entre las acciones recomendadas entre sí. Para resolver estos conflictos, la opinión que se impuso fue la del coordinador NOC.

JADE es una herramienta didáctica y configurable que aceleró el desarrollo del MAS en el presente trabajo. Sin ella, o sin sus módulos adicionales para generar servicios web o tener persistencia en bases de datos, el trabajo de construir los componentes necesarios para que el sistema funcionara, hubiera sido enorme y fuera de los límites de tiempo del trabajo de grado.

Un MAS permite, en este trabajo, reducir la complejidad de definir las interacciones y comportamiento de los agentes asociados a los dispositivos de red. Con la facilidad de definir clases de agentes diferentes y su organización en una jerarquía, se pueden modelar diferentes tipos de dispositivos así como de los comandos a ejecutar en ellos. Es por lo tanto una alternativa recomendable para soluciones en redes de comunicaciones.

El uso de principios de metodologías ágiles en la implementación como la transparencia (en el sentido de compartir un lenguaje y objetivos del sistema) y la inspección periódica de los avances, que permitió tempranamente corregir errores en la implementación del sistema (adaptación), resultan aplicables en el desarrollo de MAS.

El tiempo de diagnóstico de un incidente se reduce con el uso del sistema MAS desarrollado, debido a que la ejecución de comandos sobre los dispositivos se automatizan y no requieren de intervención humana. Así como el aplicar el conjunto de

reglas que recoge las prácticas de los ingenieros NOC, evitan a los usuarios inexpertos la demora de aplicarlas para llegar a un diagnóstico.

Mantener un sistema de diagnóstico automático de fallas en empresas de telecomunicaciones puede convertirse en una ventaja competitiva, pues debido a las exigencias regulatorias en cuanto a mantener un nivel de servicio, así como una solución rápida de incidentes, la empresa que aplique la mejor solución podrá diferenciarse de su competencia.

La metodología para el desarrollo de MAS en JADE de Nikraz et al. resultó ser, en este trabajo, una guía sencilla con entregables aplicables en el proceso de implementación. El proceso de definir los agentes, el esquema de despliegue, las interacciones entre agentes y la estructura de los mensajes de las etapas de análisis y diseño, hace que la implementación sea mucho más rápida y que se evite la reescritura de código al determinar tempranamente las interacciones y mensajes necesarios.

8.2 Recomendaciones

Para trabajos futuros el sistema se puede extender creando módulos para otras marcas de dispositivos de comunicaciones o incluso utilizando protocolos como snmp para interactuar con éstos.

Para trabajos futuros, se pueden incluir más diagnósticos en el sistema, para lo cual se deberán aumentar las reglas y los comandos a ejecutar. Si existe un cambio en el sistema operativo de los *router*, o si éstos son cambiados por otros de otras marcas, es necesario un cambio en los comandos a ejecutar y es posible que también se deba actualizar el conjunto de reglas.

El sistema se puede extender utilizando sensores de estado de los cables de fibra óptica y a su vez agentes que recojan esta información. Incluso se puede hacer un monitoreo indirecto si se establece que el estado de la fibra se puede determinar como una función del estado de los servicios que pasan por ella.

Se recomienda utilizar una clasificación jerárquica de incidentes, así como un enfoque que permita extraer y mantener el conocimiento acerca de cómo se clasifican y cómo se resuelve cada tipo de incidente.

Implementar un proceso de revisión periódica de las causas raíz, así como de los diagnósticos del sistema, porque el ingreso de nuevas plataformas y el retiro de otras, hace que la red de telecomunicaciones presente cambios y por lo tanto las causas raíz también cambiarán.

Mantener una base actualizada de servicios, es decir un sistema que unifique las bases de datos de servicios por segmentos y plataformas. En lo posible utilizar un solo CRM y no varios por los segmentos que tenga la empresa.

Adquirir sistemas de configuración automática de servicios. Esto reducirá los errores humanos así como aumentará la velocidad de atención de solicitudes de conexión y desconexión de servicios.

Anexo 1. Programa para ejecutar comandos

```
from conntelnet import Conntelnet
from parseprompt import getprompt
from parseports import parseports
from cpepro import Cpepro
import sys
import re
import pexpect
import logging
import properties
from time import gmtime, strftime


class runcmd(Conntelnet):
   def run(self, ip, usr, pwd, cmd):
      logging.basicConfig(filename=properties.base_dir+"tmp/"+ip                              +
'.log',level=logging.DEBUG)
      logging.info( strftime("%Y-%m-%d %H:%M:%S", gmtime()))
      self.cpepro = Cpepro("","",ip, "", "", "", "", "" )
      result=""
      self.ip = ip
      self.usr= usr
      self.pwd= pwd
         self.cmd= cmd
      self.verbose=False
      self.cpepro.usr=""
      self.cpepro.ping=self.pingfromos(self.ip)
      if self.cpepro.ping == True:
         result = self.execute(self.ip, self.usr, self.pwd, self.cmd)
         logging.info(result)
      else:
         result="ERROR-PING"
         self.cpepro.error=result
      return result

       def execute(self, ip, usr, pwd, cmdscript ):
      result=None
      answer=""
      self.verbose=False
      cmd="ssh "+ usr +"@" + ip
      if self.verbose:
          print "spawning: "+ cmd
```

```
      try:
         child=pexpect.spawn( cmd )
         child.timeout=12
         self.conn=child
         self.cpepro.usr=usr

         index = child.expect (['assword:', 'refused', 'yes/no', 'No route to host',
'Connection timed out' ])
         if index == 0:
            child.sendline(pwd)
            answer="OK"
            logging.info("ssh ok")
         elif index == 1:
            answer="REFUSED#"
         elif index == 2:
            child.sendline("yes")
            child.expect('assword:')
            child.sendline(pwd)
            answer="OK"
         elif index == 3:
            answer="NO-ROUTE-TO-HOST#"
         elif index == 4:
            answer="TIMEOUT#"
      except pexpect.EOF:
         answer="EOF#"
      except pexpect.TIMEOUT:
         answer="TIMEOUT#"
      if self.verbose:
          print "answer: "+answer

      if answer=="OK" :
         try:
            if self.verbose:
                print child.before + child.after
            index = child.expect(['#','>','assword:'])
            if self.verbose:
                print "index found:"
                print index

            if index == 0 or index==1:
               result = getprompt(child.before + child.after)
               self.cpepro.name=result
               self.cpepro.via="ssh"
               logging.info(" prompt found")
               child=self.conn
```

```
                        child.sendline(cmdscript)
                index = child.expect (['>', '#', 'ssword:' ])
                result = child.before + child.after
            elif index == 2:
                result = "ERROR-PASSWORD#"
                self.cpepro.error=result
                result = self.login_telnet(ip, usr, pwd)

            if self.verbose:
                print "r "+ result
            if self.verbose:
                print "Connected to "+ ip
        except pexpect.EOF:
            result="EOF#"
            self.cpepro.error="ERROR-EOF"
        except pexpect.TIMEOUT:
            result="TIMEOUT#"
            self.cpepro.error="ERROR-TIMEOUT"
    else:
        logging.info("trying telnet")
        result = "ERROR-SSH#"
    return result


def enable(self):
    self.conn.sendline("ena"  )
    self.conn.expect('Password:')
    #if self.verbose:
    #    self.showbuf()
    self.conn.sendline("radon"  )
    self.conn.expect(self.prompt)
    if self.verbose:
        #self.showbuf()
        self.setstatus("enabled")
        if self.verbose:
            print self.msgs["enabled"]
```

Bibliografía

[1] Solarwinds, «SolarWinds,» SolarWinds, 20 9 2016. [En línea]. Available: http://www.solarwinds.com/. [Último acceso: 20 9 2016].

[2] Cacti, «Cacti,» 1 10 2012. [En línea]. Available: http://www.cacti.net/. [Último acceso: 20 9 2016].

[3] Cheetah Technologies, L.P., «Cheetah Products,» Cheetah Technologies, L.P., 01 01 2016. [En línea]. Available: http://www.cheetahtech.com/. [Último acceso: 20 9 2016].

[4] E. Ardila, *Arquitectura para el manejo de congestión en una red de datos corporativa con participación del usuario, basado en inteligencia computacional.,* Bogotá, 2013.

[5] J. Castro, *Diseño de un modelo para analizar el tráfico ip generado por una red lan mediante agentes inteligentes como mecanismo para identificar comportamientos anormales en la red. Tesis de Maestría.,* Bogota: Universidad Nacional de Colombia, 2007.

[6] J. Sánchez, *Modelo de control de congestión para redes IP de área local. Tesis de Maestría.,* Bogotá: Universidad Nacional de Colombia, 2008.

[7] D. Ghosh, R. Sharman y S. Upadhyaya, «Self-healing Systems - survey and synthesis,» *Decision Support Systems,* vol. 1, nº 42, pp. 2164-2185, 2007.

[8] S. Corsava y V. Getov, «Intelligent architecture for automatic resource allocation in computer clusters,» *IPDPS: Proceedings of the 17th international symposium on parallel and distributed processing,* 2003.

[9] M. N. Huhns, V. T. Holderfield y R. L. Guitierrez, «Robust software via agent-based redundancy,» *Proceedings of the second international joint conference on autonomous agents and multiagent systems,* pp. 1018-1019, 2003.

[10] G. Tesauro, A multi-agent systems approach to autonomic computing, IBM Press, 2004, pp. 464-471.

[11] J. Gomez, «Metodologías para desarrollo de sistemas multi-agente,» *Revista Iberoamericana de Inteligencia Artificial,* vol. 1, nº 18, pp. 51-63, 2003.

[12] Tropos, «Methology related papers,» [En línea]. Available: http://www.troposproject.org/view/Methodologypapers. [Último acceso: 19 Junio 2016].

[13] F. Bellifemine, A. Poggi y G. Rimasa, «Developing multi-agent systems with JADE,» *Lecture Notes in Artificial Intelligence,* vol. 1571, pp. 29-42, 2001.

[14] S. Deloach y J. Garcia, «O-MaSE: a customizable approach to designing and building complex adaptative multi-agent systems,» *Agent-oriented Software Engineering,* vol. 4, nº 3, 2010.

[15] C. Iglesias, M. Garijo, J. González y J. Velasco, Analysis and design of multiagent systems using MAS-CommonKADS., Madrid: Universidad Politécnica de Madrid, 1998.

[16] M. Wooldridge, An Introduction to Multiagent Systems, Liverpool, UK: John Wiley & Sons, ltda, 2002.

[17] Maryland University, «What is KQML,» Computer Science, 2014. [En línea]. Available: http://www.csee.umbc.edu/csee/reseach/kqml/. [Últimо acceso: 20 Junio 2016].

[18] Foundation for Intelligent Physical Agents (FIPA), «Interaction Protocol Specifications,» 2002. [En línea]. Available: http://www.fipa.org/repository/ips.php3. [Último acceso: 04 04 2015].

[19] MIT Media Lab, «Aglets Software Development Kit,» [En línea]. Available: https://sourceforge.net/projects/aglets/files/. [Último acceso: 17 Enero 2017].

[20] «JACK, AOS (Autonomous Decision-Making software),» [En línea]. Available: http://aosgrp.com/products/jack. [Último acceso: 20 06 2016].

[21] «Grasshopper Platform,» Community Research and Development Information Service, [En línea]. Available: http://cordis.europa.eu/infowin/acts/analysys/products/thematic/agents/ch4/ch4.htm (. [Último acceso: 20 Junio 2016].

[22] «FIPA-OS Toolkit,» [En línea]. Available: http://sourceforge.net/projects/fipa-os/?source=directory. [Último acceso: 20 Junio 2016].

[23] M. Wooldrige y H. Nwana, Software Agent Technologies, BT Technology, 2006.

[24] J. Jarvis, R. Ronquist, D. Jarvis y C. Lakhmi, «Holonic Execution: A BDI Approach.,»

Studies in Computational Intelligence., vol. 106, 2008.

[25] A. W. Colombo, «An agent-based intelligent control platform for industrial holonic manufacturing systems,» *IEEE Transactions on Industrial Electronics,* vol. 53, nº 1, pp. 322-337, 2006.

[26] D. Z. Zhang, A. Anosike y M. K. Lim, «Dinamically integrated manufacturing systems (DIMS)---a multi-agent approach,» *IEEE Transactions on systems, man and cybernetics,* vol. 37, nº 5, pp. 824-850, 2007.

[27] E. M. Davidson, S. D. McArthur, J. R. McDonald, T. Cumming y I. Watt, «Applying multi-agent system technology in practice: automated management and analysis of SCADA and digital fault recorder data,» *IEEE Transacions on power systems,* vol. 21, nº 2, pp. 559-567, 2006.

[28] S. D. McArthur, S. M. Strachan y G. Jahn, «The design of a multi-agent transformer condition monitoring system,» *IEEE Transactions on power systems,* vol. 19, nº 4, pp. 1845-1852, 2004.

[29] D. P. Buse y Q. H. Wu, «Mobile agents for remote control of distributed systems,» *IEEE Transactions on industrial electronics,* vol. 51, nº 6, pp. 1142-1149, 2004.

[30] J. Zhou, G. Chen, W. Zhang, W. Yan y Q. Chen, «Multi-agent based distributed monitoring and control of hazard installations,» *Proceedings of the 2nd IEEE Conference on Industrial Electronics and Applications,* pp. 2892-2895, 2007.

[31] J. Galdun, L. Tak'a'c, J. Ligu's, J. M. Thirie y J. Samovsky, «Distributed control systems reliability: consideration of multi-agent behavior,» *Proceedings of the 6th International Symposium on Applied Machine Intelligence and Informatics,* pp. 157-162, 2008.

[32] K. Fregene, D. C. Kennedy y D. W. L. Wang, «Toward a systems and control-oriented agent framework,» *IEEE Transactions on Systems, Man and Cybernetics,* vol. 35, nº 5, pp. 999-1012, 2005.

[33] K. Hwang, S. W. Tan, M. C. Hsiao y C. S. Wu, «Cooperative multi-agent congestion control for high-speed networks,» *IEEE Transactions on Systems, Man and Cybernetics,* vol. 35, nº 2, pp. 255-268, 2005.

[34] D. Srinivasan y M. C. Choy, «Cooperative multi-agent system for coordinated traffic signal control,» *IEEE Proceedings: Intelligent Transport Systems,* vol. 153, nº 1, pp. 41-50, 2006.

[35] T. Nagata y H. Sasaki, «A multi-agent approach to power system restoration,» *IEEE Transactions on Power Systems,* vol. 17, nº 2, pp. 457-462, 2002.

[36] S. E. Widergren, J. M. Roop, R. T. Guttromson y Z. Huang, «Simultating the dynamic coupling of market and physical system operations,» *Proceedings of IEEE Power Engineering Society General Meeting,* vol. 1, pp. 748-753, 2004.

[37] D. Koesrindartoto, J. Sun y L. Tesfatsion, «An agent-based computational laboratory for testing the economic reliability of wholesale power market designs,» *IEEE Proceedings on Power Engineering Society General Meeting,* vol. 3, pp. 2818-2823, 2005.

[38] A. L. Dimeas y N. D. Hatziargyriou, «Operation of a multi-agent system for microgrid control,» *IEEE Transactions on Power Systems,* vol. 20, nº 3, pp. 1447-1455, 2005.

[39] C. Heinze, S. Goss, T. Josefsson, K. Bennett, S. Waugh, I. Lloyd, G. Murray y J. Oldfield, «Interchanging agents and humans in military simulation,» *AI Magazine,* vol. 23, nº 2, pp. 37-48, 2002.

[40] S. Karim y C. Heinze, «Experiences with the design and implementation of an agent-based autonomous UAV controller,» de *Proceedings of the 4th international Joint Conference on Autonomous Agents and Multi-Agent Systems*, Utrecht, The Netherlands, 2005.

[41] C. de Pablos, J. J. Lopez-Hermoso, S. Martin-Romo y S. Medina, Direccion y Gestión de los sistemas de información en la empresa: una vision integradora, Madrid: ESIC Editorial, 2008.

[42] D. R. Ferreira y M. M. Silva, «Using process mining for ITIL assessment: a case study with incident management,» 2008.

[43] F. Niessink y H. Van Vliet, «Towards Mature IT Services,» *Software Process - Improvement and Practice,* 1988.

[44] J. Von Bon, A. De Jong y A. Kolthof, IT Service Management - An Introduction, Zaltbommel, Nederlands: Van Haren Publishing, 2007.

[45] Cabinet Office;, ITIL Service Operation, London, UK: The Stationary Office, 2011.

[46] K. Saarelain y M. Jantti, «Creating an ITIL based multidimensional incident analytics method,» de *ICONS 2015 : The Tenth International Conference on Systems*, Helsinki, 2015.

[47] J. Oksanen, Organizing a Network Operation Center On Campus, Helsinki: Terena, 2013.

[48] A. Latrache, E. H. Nfaoui y J. Boumhidi, «Multi agent based incident management system according to ITIL,» *Intelligent Systems and Computer Vision,* pp. 1-7, 2015.

[49] Bmc products, Footpring IT service management for mid-size organization software, 2009.

[50] Remedy Service Desk software for incident management, 2009.

[51] NetApplications, «Net Marketshare,» NetApplications, 03 11 2016. [En línea]. Available: https://netmarketshare.com/. [Último acceso: 03 11 2016].

[52] Y. Penga, Y. Zhanga y Y. Tangb, «An incident information management framework based on data integration, data mining, and multi-criteria decision making,» *Decision Support Systems jornal,* vol. 51, nº 2, pp. 316-327, 2011.

[53] S. Franklin y A. Graesser, «Is it an agent or just a program?,» de *Proceedings of the 3rd International Workshop Agent Theories, Architectures, and Languages*, New York, NY, 1996.

[54] M. Wooldridge y N. R. Jennings, «Intelligent agents: theory and practice,» *The Knowledge Engineering Review,* vol. 10, nº 2, pp. 115-152, 1995.

[55] M. Wooldridge y N. R. Jennings, Application of intelligent agents, New York, NY, USA: Springer, 1998.

[56] D. Weyns, Architecture-based design of multi-agent systems, Berlin: Springer-Verlag, 2010.

[57] C. Kim On, G. Kim Soon, A. Rayner, A. Patricia y L. Dickson, «Agent Architecture: An Overview,» *Transactions on science and technology,* vol. 1, nº 1, pp. 18-35, 2014.

[58] S. Millet y N. Tune, Patterns, Principles, and Practices of Domain-Driven Design, New Jersey: Wrok, 2013.

[59] S. Poslad, «Specifying Protocols for Multi-Agent Systems Interaction,» *ACM Transactions on Autonomous and Adaptive Systems,* vol. 2, nº 4, pp. 15-24, 2007.

[60] R. H. Shumway y D. S. Stoffer, Time Series Analysis and Its Applications, New York, USA: Springer, 2011.

[61] G. J. Kerns, Introduction to Probability and Statistics using R, Youngstown, Ohio: Youngstown University, 2010.

[62] R. Foundation, «R project,» R Core Group, 06 06 2016. [En línea]. Available: https://www.r-project.org/about.html. [Último acceso: 24 02 2016].

[63] R. S. Pressman, Software Engineering, Boston: McGrawHill, 2001.

[64] Statista, «Statista The Statistics Portal,» Statista, 03 03 2016. [En línea]. Available: https://www.statista.com/statistics/235289/global-ethernet-switch-revenue-market-share-by-vendors/. [Último acceso: 01 04 2017].

[65] Cisco, «About Cisco,» 01 01 2017. [En línea]. Available: http://www.cisco.com/c/en/us/about.html. [Último acceso: 01 04 2017].

[66] M. Nikraz, G. Caire y P. Bahri, «A methodology for the analysis and design of multi-agent systems using JADE,» *International Journal of Computer Systems Science & Engineering,* p. 41, 2006.

[67] American National Standard Institute, «American National Standard T1.523-2001,» 2001. [En línea]. Available: http://www.atis.org/glossary/default.aspx. [Último acceso: 04 04 2015].

[68] P. Krunchten, «Architectural Blueprints—The “4+1” View,» *IEEE software,* vol. 12, nº 6, pp. 42-50, 1995.

[69] The Eclipse Foundation, «Eclipse,» The Eclipse Foundation, 30 Octubre 2016. [En línea]. Available: http://www.eclipse.org/. [Último acceso: 03 Noviembre 2016].

[70] A. Knaeuper y W. B. Rouse, «A Rule-Based model of human problem-solving behavior in dynamic environments,» *IEEE Transactions on Systems, man and cybernetics,* Vols. %1 de %2SMC-15, nº 6, pp. 708-720, 1985.

[71] M. B. Hassine, «Easy Rules Java Rule Engine,» 08 Febrero 2017. [En línea]. Available: http://www.easyrules.org/. [Último acceso: 30 Abril 2017].

[72] http://www.jboss.org/drools/team.html, «Drools Expert User Guide,» 01 06 2016. [En línea]. Available: https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf. [Último acceso: 01 04 2017].

[73] Scrum organization, «Scrum.org,» Scrum organization, 01 11 2016. [En línea]. Available: https://www.scrum.org. [Último acceso: 03 11 2016].

[74] Telecom Italia, «Jade Site,» Telecom Italia, 10 5 2016. [En línea]. [Último acceso: 15 10 2016].

[75] Junit team, «JUnit,» Junit Team, 18 Abril 2016. [En línea]. Available: http://junit.org. [Último acceso: 03 Nov 2016].

[76] Smartbear, «Soap UI,» Smartbear, 10 11 2016. [En línea]. Available: https://www.soapui.org/. [Último acceso: 13 11 2016].

[77] GNS3 Technologies Inc., «GNS3,» GNS3 Technologies Inc., 5 5 2016. [En línea]. Available: https://www.gns3.com/community. [Último acceso: 10 5 2016].

[78] M. C. Valiente, Improving IT Service Management using an Ontology-Based and Model Driven Approach, Alcalá de Henares: Universidad de Alcalá, 2011.

[79] E. Sánche-Nielsen, A. Padrón-Ferrer y F. Marreo-Estévez, «A multi agent system for incident management solutions on IT infrastructures,» *Advances in Artificial Intelligence Lecture notes in computer science,* vol. 7023, pp. 22-31, 2011.

[80] Net support service desk software for incident management, 2010.