UNIVERSIDAD NACIONAL DE COLOMBIA

Modelo Basado en Aprendizaje de Máquinas para el Manejo de Riesgo de Falla Durante la Composición de Servicios Web

Byron Enrique Portilla Rosero

Universidad Nacional de Colombia
Facultad de Minas, Escuela de Sistemas
Medellín, Colombia
2011

Modelo Basado en Aprendizaje de Máquinas para el Manejo de Riesgo de Falla Durante la Composición de Servicios Web

Byron Enrique Portilla Rosero

Tesis presentada como requisito parcial para optar al título de:
Magister en Ingeniería de Sistemas

Director:
Ph.D., Jaime Alberto Guzmán Luna

Línea de Investigación:
Inteligencia Artificial
Grupo de Investigación:
SINTELWEB

Universidad Nacional de Colombia
Facultad de Minas, Escuela de Sistemas
Medellín, Colombia
2011

Con gratitud y amor dedico a mis padres y a mi hermana Carmencita este trabajo producto de mi permanente lucha por aprender a construir conocimiento porque ellos, han sido mi fortaleza y apoyo en las horas difíciles de mi vida estudiantil. Gracias.

Agradecimientos

Agradezco la doctor Jaime Alberto Guzmán Luna, director de mi tesis, quien con sus consejos, críticas, orientación y colaboración decidida me guió en el proceso de mi formación investigativa; asimismo, por darme la oportunidad de ser uno más de sus estudiantes.

De igual manera, agradezco a los jurados, Ph,D. Giner Alor Hernández y Ph.D. José Nelson Pérez Castillo por los conceptos emitidos en la corrección de mi tesis los cuales fueron oportunos, significativos y de gran ayuda en el mejoramiento de la misma.

Resumen

Los trabajos realizados hasta el momento en cuanto a la composición de servicios Web, se caracterizan en su mayoría por la utilización de modelos, donde las especificaciones de los servicios Web utilizadas para generar un plan de composición, están totalmente definidas. Es así como mecanismos orientados a la composición que busquen minimizar el riesgo de fallas además de tener la información funcional de cada servicio, tienen la información pertinente al factor de riesgo de que tales servicios fallen; esto, con el fin de guiar al mecanismo de composición para seleccionar la mejor solución entre todas las posibles composiciones que minimice las posibilidades de que estas fallen. Desafortunadamente la tarea de calificar el factor de riesgo requiere de expertos para su evaluación o en el peor de los casos esta información parte de los propietarios de los propios servicios con lo cual el factor de veracidad de dicha información no es la mejor. Es así como el problema principal en este proceso de composición, es obtener estas especificaciones asociadas al riesgo de falla. En consecuencia, este trabajo de tesis plantea una solución al manejo de riesgos de fallas durante la composición con servicios Web, donde a través del aprendizaje de máquinas, se adquiera de manera automática y bajo el factor de certeza propio del compositor cada uno de los criterios relacionados con el factor de riesgo, que permitan identificar, cuáles serán los mejores servicios que harán parte de este proceso de composición y mejorar de esta forma su tarea de composición de servicios. En esta tesis particularmente se trabajaron tres factores de riesgos de falla que no se trataron previamente en el estado del arte como lo son: la disponibilidad y reactividad de los servicios bajo periodos de tiempo discriminados en días y horas de la semana y el manejo de creencias.

Palabras clave: servicios Web, composición de servicios Web, aprendizaje automático, manejo de riesgos.

Abstract

The researches carried out up to now for the Web services composition, are characterized mostly by the use of models, where the specifications of the Web services used to generate a plan of composition, are completely defined. It is as well as mechanisms oriented to the composition that seek to minimize the risk of failures besides of having the functional information of each service, have the information concerning the factor of risk that such services fail; this, in order to guide to the mechanism of composition to select the best solution among all the possible compositions that minimize the possibilities that these fail. Unfortunately the task to qualify the factor of risk requires of experts for its evaluation, or if worse comes to worst, this information comes from the owners of the own services in which the factor of truth of this information is not the best.

Therefore, the main problem in this composition process is to obtain these specifications related to the risk of failure. Consequently, this thesis work points out a solution to the management of failure risks during the composition with Web services, where through the machines learning, it is possible to acquire in an automatic way and under the factor of certainty of the composer, each criteria related to the factor of risk, that allow to identify, which will be the best services than will be part of this process of composition, and therefore, improving their services composition task.

Particularly, three failure risks factors were worked in this thesis. These factors have not been treated previously in the state of the art. They are: the availability and reactivity of the services under periods of time discriminated in days and hours of the week and the management of beliefs.

Keywords: Web services, Web services composition, machine learning, risk analysis.

Contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

Lista de Abreviaturas

Abreviaturas

Abreviatura	Término
AI	Inteligencia Artificial
CBR	Case Based Reasoning
IBM	International Business Machine
ILP	Inductive logic programming
MDP's	Markov decision process
OWL	Ontology Web Language
OWL - S	Ontology Web Language for Services
PROLOG	PROGrammation en LOGique
QoS	Quality of Service
SOAP	Simple Object Access Protocol
SW	Servicios Web
SWS	Semantic Web Services ó Servicios Web Semánticos
TDIDT	Top Down Induction Decision Trees
TILDE	Top-down Induction of Logical Decision Trees
UDDI	Universal Description Discovery and Integration
W3C	World Wide Web Consortium
WS-CDL	Web Services Choreography Description Language
WS-COORDINATION	Web Services Coordination

Abreviatura	Término
WS-SECURITY	Web Services Security
WS-TRANSACTION	Web Services Transaction
BEPL	Business Process Execution Language
WSCI	Web Service Choreography Interface
WSCL	Web Services Conversation Language
WSDL - S	Web Service Description Language
WSMO	Web Services Modeling Ontology
WSFL	Web Services Flow Language

Introducción

La evolución de la Web, ha cambiado el paradigma de como las personas se comunican y realizan sus actividades diarias de negocio, trabajo, ocio entre otros de manera ágil y con múltiples recursos a su disposición en diferente presentación y formato. Ante esta evolución, emergen nuevas tecnologías que se adaptan cada vez más a los requerimientos de los usuarios con el objetivo de satisfacerlos. Dentro de esas tecnologías, se encuentra un conjunto de aplicaciones modulares y multiplataforma conocidas como servicios Web (SW) las cuales, soportan una o más utilidades que satisfacen una serie de requerimientos solicitados por un cliente.

Estos servicios, utilizan un conjunto de descripciones y reglas que les permiten su interacción en la Web usando lenguajes de descripción (WSDL), protocolos de acceso (SOAP), entre otros. No obstante, la misma evolución de la Web ha influido en el mejoramiento de los SW apuntando a un desarrollo semántico en el cual, los servicios se han desarrollado con descripciones semánticas (SWS) de manera que permitan mejorar su descubrimiento, invocación e interoperabilidad y su interacción con las personas es decir, solicitar los requerimientos sin tener que indicar de donde se obtiene o qué significado tiene cada requerimiento.

Un aspecto importante en este ambiente de servicios es la composición de servicios Web. Este, es un proceso que incluye un análisis cuidadoso de los requerimientos, la semántica y el comportamiento de los servicios existentes, la verificación de servicios, la adaptación, la contratación y su manejo. Cuando uno de los requerimientos solicitados por un cliente no es posible ser satisfecho por un servicio existente, se realiza este proceso construyendo un nuevo servicio, integrando servicios ya existentes y así obtener los requerimientos no satisfechos anteriormente. Para ello, se ha utilizado diferentes tecnologías como la planificación en IA, cálculos de situaciones o procesos de Markov entre otras.

Sin embargo, en el campo de la composición, los problemas aparecen cuando los servicios utilizados dentro de este proceso, conllevan a fallas inesperadas ocasionadas por el entorno de los servicios o el compositor.

Con el fin de valorar dichas fallas, se recurrió a una monitorización constante del comportamiento de los servicios durante el proceso de ejecución haciendo uso de los parámetros de calidad de los servicios (QoS) ya que estos contemplan información adicional a la propia funcionalidad de los servicios permitiendo hacer un análisis multi-dimensional de dicho comportamiento desde el punto de vista de estos parámetros de calidad.

Estos parámetros, permiten representar y analizar el comportamiento de los servicios en cuanto al entorno en el que se ejecuta y al entorno de composición en el cual su

participación se requiere. De esta manera, es posible llevar a cabo composiciones que tengan en cuenta el modelado de los posibles riesgos de falla que se presentan en dicha composición. Esta evaluación, hace posible la identificación de servicios más adecuados para cumplir tareas en particular bajo la óptica de estos parámetros de calidad.

Igualmente, se analiza el comportamiento del compositor a través de la monitorización de las creencias que este asume como verdaderas para cada servicio, de manera que se identifique la información que más se aproxima a la que los servicios generan en el mundo real y de esta forma disminuir el riesgo de falla ocasionado por el compositor.

Teniendo en cuenta lo anterior, esta tesis se enfoca en el manejo de riesgos de falla en composiciones de servicios Web mediante la utilización de parámetros de calidad con los cuales se evalúan dos aspectos importantes:

La minimización de los fallos de los servicios en relación a su entorno, lo que implica una mala selección del servicio a ejecutar en la composición.
La minimización de las fallas producidas en la composición de servicios relacionadas a la mala suposición del compositor al elegir las instancias de los servicios a ejecutar.

En primer lugar, el uso de los QoS representa el conjunto de parámetros que describen el comportamiento habitual de los servicios de forma que se analice aspectos relevantes del comportamiento de los servicios y, que permitan realizar un análisis de riesgos de falla asociados al ambiente dinámico de la composición. En segundo lugar, la información utilizada en procesos de composición de servicios, está directamente influenciada por el entorno en el momento actual que se realiza el proceso; es decir, al ser la Web un entorno dinámico, la información contenida en ésta, está sujeta a cambios; por lo tanto, los datos son representativos en ese momento y lugar para tomar las decisiones en cuanto al manejo de riesgos. De manera que su representación esté acorde al comportamiento de los servicios durante su ejecución.

En algunas investigaciones encontradas en cuanto al tema de riesgos (Liu, Zhang, Ren, Liu, & Zhang, 2009), (El Haddad, Manouvrier, Ramirez, & Rukoz, 2008), (Cardoso, Sheth, Miller, Arnold, & Kochut, 2004) explicadas detenidamente en el capítulo 2 de esta tesis, se identificó que la información asociada a cada parámetro utilizado para hacer el análisis de riesgos se asigna por un experto o en algunos casos suministrada por los proveedores de los mismos servicios lo cual implica que la veracidad de esta información no esté acorde al comportamiento dinámico de los servicios y el compositor. Por lo tanto, esta tesis se centra en la generación de un modelo de aprendizaje de máquinas, con el fin de adquirir las especificaciones necesarias de QoS, que permitan obtener la información relacionada con el riesgo de falla de las composiciones de servicios.

Objetivos de la tesis

Objetivo general

Proponer un modelo basado en aprendizaje de máquinas, que permita de manera automática adquirir el conocimiento relacionado con el factor de riesgo que presentan los servicios que participan en una composición, considerando su comportamiento en el servicio Web final como un todo, con el fin de minimizar el riesgo de falla.

Objetivos específicos

- Definir los criterios y sus métricas relacionados con el factor de riesgo de falla, que permitan guiar el proceso de composición de servicios Web.

- Evaluar y seleccionar las técnicas de aprendizaje de máquinas que hagan posible la adquisición de manera automática del conocimiento asociado al factor de riesgo de falla de las composiciones de servicios.

- Proponer un modelo de aprendizaje de máquinas a partir de las técnicas analizadas, donde sea posible obtener de manera automática la información relacionada con los riesgos de falla de las composiciones de servicios.

- Proponer un mecanismo general que permita integrar el modelo de aprendizaje de máquinas propuesto con técnicas de planificación orientadas a la composición de servicios.

- Validar el modelo de aprendizaje propuesto con la implementación de un prototipo para un dominio específico y realizar una comparación con otros modelos con base en los criterios definidos.

Metodología

La metodología que se empleó en esta tesis, consta de dos aspectos principales; los cuales representan la parte teórica y la parte estructural respectivamente. En la primera, se argumenta cada uno de los conceptos necesarios para la adquisición de la información pertinente, en la valoración de reglas y fórmulas que describen los criterios y métricas a evaluar; los cuales, están directamente relacionados con el manejo de fallas en el proceso de composición de servicios. Seguidamente, se realiza el análisis y selección de las técnicas de aprendizaje de máquinas que permitan el aprendizaje automático de los criterios asociados al factor de riesgo de falla de las composiciones de servicios.

Una vez se culminó esta fase, se procedió con la fase de estructuración en la cual, se proyecta el modelo de aprendizaje de máquinas dentro del proceso de composición de servicios que posteriormente, se integró con la técnica planificación orientada a la composición de servicios Web y, finalmente, se evaluó y validó con la implementación de un prototipo teniendo en cuenta los criterios definidos.

En detalle la metodología seguida en este trabajo de tesis se dividió en cinco etapas principales, agrupadas en dos fases asociadas con la parte de investigación teórica y la parte de investigación estructural respectivamente.

En la tabla 1, se presenta la relación existente entre cada una de las fases de la metodología y sus respectivas etapas con los objetivos específicos propuestos. De esta manera se garantiza el cumplimiento de los objetivos.

A continuación, se explica brevemente cada una de las etapas y sus respectivas actividades.

Tabla 1: Relación entre la metodología y los objetivos específicos

Metodología		Objetivos específicos
Fases	Etapas	
Fase 1	Etapa 1	Objetivo 1
	Etapa 2	Objetivo 2
Fase 2	Etapa 3	Objetivo 3
	Etapa 4	Objetivo 4
	Etapa 5	Objetivo 5

Fase 1:

Etapa 1: Identificación de los criterios y sus métricas relacionados con los riesgos de falla que permitan guiar el proceso de composición de servicios Web.

Se hizo una revisión bibliográfica en busca de los parámetros de servicios Web que evalúan los riesgos en la composición de servicios y que son considerados en la literatura como relevantes en dicho proceso.

Posteriormente se definió tres criterios: disponibilidad, reactividad y manejo de creencias los cuales, representan el comportamiento de la composición de servicios desde dos puntos de vista distintos. El primero sobre el comportamiento de los servicios y el segundo sobre el comportamiento del sistema de composición.

Finalmente se determinó el conjunto de métricas que representa a cada uno de los criterios definidos y se calculó la fórmula que evalúa el riesgo de falla para cada uno de los servicios.

El análisis completo de esta etapa se describe en el capítulo 2.

Etapa 2: Evaluación y selección de las técnicas de aprendizaje de máquinas que permitan el aprendizaje automático de los criterios asociados al factor de riesgo de falla en las composiciones de servicios.

Mediante la búsqueda bibliográfica se determinó aquellas técnicas de aprendizaje de máquinas más utilizadas en la composición de servicios; a partir de esta actividad, se realizó una evaluación de cada una de las técnicas identificadas, teniendo en cuenta las fortalezas y debilidades de cada una con respecto a las otras.

Finalmente, se seleccionó la técnica de aprendizaje de máquinas que mejor describió el aprendizaje automático de los criterios definidos en el capítulo 2.

El análisis de estas técnicas de aprendizaje es descrito en el capítulo 3 de esta tesis, donde se hace una comparación de las técnicas utilizadas para la composición de servicios.

Fase 2:

Etapa 3: Proposición de un modelo de aprendizaje de máquinas, que de manera automática obtenga la información relacionada con los riesgos de falla de las composiciones de servicios, para generar planes de composición que se vayan refinando con el tiempo y la experiencia del sistema.

Se planteó una arquitectura de aprendizaje de máquinas para la adquisición de la información relacionada con el factor de riesgo de los servicios durante su composición.

Se diseñó el algoritmo de aprendizaje de máquinas que contempla los criterios definidos en la etapa 2 y la técnica de aprendizaje obtenida en la etapa 2.

En el capítulo 4, se muestran los resultados de esta actividad.

Etapa 4: Proposición de un mecanismo general para integrar el modelo de aprendizaje de máquinas propuesto con técnicas de planificación orientadas a la composición de servicios Web.

Se desarrolló la implementación de la arquitectura de aprendizaje descrita en la etapa 3.
Se evaluó la representación de los servicios por parte de las técnicas de planificación orientadas a la composición de servicios y se definió una caracterización general para representar un problema de composición de servicios.

Se desarrollaron dos traductores que permiten la integración del modelo de aprendizaje de máquinas con el sistema de composición de servicios web Semánticos.

Los resultados de las actividades de esta etapa están detallados en el capítulo 5.

Etapa 5: Validación del modelo de aprendizaje propuesto:

Se diseñó un ambiente de pruebas donde se valida el modelo propuesto resultado de la etapa 4., a partir de simulaciones y datos reales utilizando un dominio de compras por internet.

El ambiente de pruebas y el análisis de los resultados son expuestos en el capítulo 6.

Contribuciones principales alcanzadas

- La principal contribución de esta tesis, es el desarrollo de un modelo de aprendizaje de máquinas para el manejo de riesgos de falla durante la composición de servicios web, haciendo uso de tres criterios que evalúan el comportamiento del servicio y el comportamiento del sistema de composición de servicios.

    - ✓ Disponibilidad, analiza si un servicio Web está disponible a un requerimiento asociado a periodos por día y hora a lo largo de la semana.
    - ✓ Reactividad de los servicios, identifica el comportamiento reactivo de los servicios afectado por restricciones de tiempo de ejecución dentro de un límite temporal discriminado por días y horas.
    - ✓ Manejo de Creencias, evalúa el desempeño del mecanismo de composición al realizar una suposición de las instancias requeridas para instanciar un servicio Web y ejecutarlo.

- La segunda contribución es el desarrollo del prototipo DOMINIUS-BP, el cual permite analizar la información de los servicios a partir de sus QoS y aprender el conocimiento de control a partir de estos datos, con el objeto de generar la representación del riesgo de falla en la composición de servicios Web.

- El conocimiento de control, es un conjunto de reglas de aprendizaje expresadas a través de reglas lógicas condicionales que resultan de aplicar la técnica de inducción de árboles de decisión lógicos, al conjunto de información adquirida en ejecución.

- La tercera contribución es el desarrollo de dos traductores que permiten integrar el prototipo de aprendizaje DOMINIUS-BP, con una técnica de planificación orientada a la composición de servicios Web, como lo es el sistema de composición "INDYGO".

    - ✓ El primer traductor, es el encargado de obtener la información de ejecución del servicio y traducirla a un lenguaje lógico de representación de datos de entrenamiento que son interpretados por el prototipo de aprendizaje DOMINIUS-BP.
    - ✓ El segundo traductor, traduce las reglas generadas por el prototipo de aprendizaje DOMINIUS-BP para los criterios de disponibilidad y reactividad, a una descripción OWL-S del servicio.

- La cuarta contribución es la extensión de la ontología OWL-S a una representación ontológica de calidad.

- La quinta contribución es la elaboración de un ambiente de pruebas haciendo uso de un conjunto de servicios descritos en las ontologías OWL-S y OWL los cuales, describen QoS para el análisis del riesgo de falla.

Difusión de resultados

Los resultados de esta tesis se presentaron en distintas revistas y congresos nacionales.

Revistas nacionales

Modelo de aprendizaje de máquinas para reducir las fallas de instanciación en composiciones de servicios. Byron Enrique Portilla Rosero, Jaime Alberto Guzmán Luna, En: Colombia Revista Politécnica ISSN: 1900-2351 ed: Politécnico Colombiano Jaime Isaza Cadavid v.9, p.58, 2009.

Una Arquitectura de Aprendizaje para Disminuir el Riesgo de Falla de Servicios Web Durante su Composición. Byron Enrique Portilla Rosero, Jaime Alberto Guzmán Luna, En: Colombia, Revista Colombiana De Tecnologías De Avanzada ISSN: 1692-7257 v.2 fasc.16 p.93, 2009.

Un Análisis al Estado del Arte en el Manejo de Riesgos de Falla Durante la Composición de Servicios Web Parcialmente Especificados. Byron Enrique Portilla Rosero, Jaime Alberto Guzmán Luna, En: Colombia Revista Colombiana De Tecnologías De Avanzada ISSN: 1692-7257 v.2 fasc.12 p.125, 2008.

Arquitectura de aprendizaje para el manejo de riesgo de falla en ambientes de composición de servicios Web. Byron Enrique Portilla Rosero, Jaime Alberto Guzmán Luna, En: Colombia Avances en Sistemas e Informática ISSN: 1657-7663 ed: Universidad Nacional De Colombia Sede Medellín, v.6 fasc.2 p.143 - ,2008.

Congresos Nacionales

Un Análisis al Estado del Arte en el Manejo de Riesgos de Falla Durante la Composición de Servicios Web Parcialmente Especificados. Byron Enrique Portilla Rosero, Jaime Alberto Guzmán Luna, VI Congreso Internacional de Electrónica y Tecnologías de Avanzada, Pamplona, Colombia, 2008.

Una Arquitectura de Aprendizaje para Disminuir el Riesgo de Falla de Servicios Web Durante su Composición. Byron Enrique Portilla Rosero, Jaime Alberto Guzmán Luna, VII Congreso Internacional de Electrónica y Tecnologías de Avanzada, Pamplona, Colombia, 2009.

1. Marco teórico y estado del arte

Este capítulo, expone el marco teórico y un análisis al estado del arte orientado a los procesos de evaluación de riesgos de falla en la composición de servicios teniendo en cuenta el uso de QoS.

1.1 Los servicios web

En la actualidad, la masificación del internet permite el crecimiento de nuevas tecnologías Web que apoyan a sus usuarios en el desarrollo de actividades educativas, industriales, económicas, de entretenimiento entre otras; haciendo de éstas, acciones interactivas y de fácil manipulación para los usuarios. Una de estas tecnologías son los servicios Web, su concepto está ligado a una aplicación ejecutada en un entorno Web.

En la literatura, existen distintas definiciones relacionadas con un servicio Web. En una primera, estos se definen como un conjunto de aplicaciones modulares, autónomas, auto-descritas que son publicadas, localizadas e invocadas a través de la Web y, que por medio del intercambio de sus datos, proporcionan un conjunto de servicios (Rao & Su, 2005). Estos, los solicitan los usuarios a través de peticiones realizadas a los proveedores de servicios los que a su vez generan procedimientos remotos, que proveen un conjunto de información dinámica a los usuarios, a través de la Web.

Estándares internacionales como IBM y W3C no se alejan de esta concepción, definiendo un servicio web como un sistema empaquetado reutilizable que interactúa con diferentes máquinas bajo distintas plataformas a través de un lenguaje de descripción WSDL (Christensen, Curbera, Meredith, & Weerawarana, 2001) y operado por el intercambio de mensajes por medio de protocolos SOAP (Snell, Tidwell, & Kulchenko, 2001).

Por lo tanto, un servicio Web es considerado como un sistema que permite una interacción heterogénea entre sistemas, los cuales son manipulados en la Web a partir de protocolos de comunicaciones y su interacción responde a solicitudes hechas por los clientes de manera que un cliente es visto como uno o un conjunto de servicios.

Una de las evoluciones de los servicios es la de alcanzar una compresión semántica por parte de las máquinas que las utilizan, por lo tanto se han propuesto los servicios Web Semánticos (SWS), los cuales son servicios Web, cuyas descripciones internas y externas están en un lenguaje que tiene semánticas bien definidas, interpretables por las máquinas (McIlraith, Son, & Zeng, 2001). Algunos de estos lenguajes son: OWL-S (Martin, y otros, 2004), WSMO (Roman, y otros, 2005) ó WSDL-S (Akkiraju, y otros, 2005).

1.2 Composición de servicios

La composición de servicios resulta de la no satisfacción de un servicio para dar respuesta a determinados requerimientos, esto se traduce a que muchos de los servicios que se encuentran en la web cumplen una función limitada y no alcanza a satisfacer las peticiones de un cliente. Es así como la composición de servicios surge para dirimir este problema (Barreiro, Albers, & Hao, 2006), siendo un aspecto básico de la computación orientada a los servicios (Qian, Lu, & Lie, 2005), que soporta la creación de servicios complejos y utiliza modelos que le permiten la composición flexible de sistemas heterogéneos a través de protocolos básicos de los servicios Web SOAP, WSDL, UDDI (Walsh, 2002).

La composición de servicios Web, es vista entonces, como el proceso de desarrollar servicios personalizados a través de otros ya existentes por medio de procesos de descubrimiento dinámico, integración y ejecución, con el fin de satisfacer los requerimientos del usuario (Chakraborty, Perich, Joshi, Finin, & Yesha, 2002). Para ello, la composición de servicios se apoya en técnicas como la coreografía y la orquestación. La primera, hace referencia al manejo de los mensajes de forma múltiple entre servicios, describiendo especificaciones lógicas temporales entre las actividades y utiliza lenguajes como WS-CDL (Kavantzas, Burdett, Ritzinger, Fletcher, Lafon, & Barreto, 2005), WSCI (Brogi, Canal, Pimentel, & Vallecillo, 2004) que hacen posible la manipulación y comprensión de los servicios, con el fin de ser interpretados. Por otra parte, se encuentra la orquestación, que actúa sobre la ejecución de determinados procesos de negocios; su interacción ocurre a nivel del mensaje, e incluye lógica de negocios y ejecución de tareas. Uno de los requerimientos de la orquestación es tener un servicio compuesto especificado completamente. Uno de los lenguajes utilizados para la definición de procesos que se ejecutan en un motor de orquestación es BPEL (Juric, Mathew, & Sarang, 2004). De igual forma, existen otros lenguajes utilizados en el proceso de orquestación que permiten modelar el flujo de información de los servicios. Estos lenguajes son: WSCL (Banerji, y otros, 2002), XLANG (Peltz, 2003), WSFL (Bussler, Hull, McIlraith, Orlowska, Pernici, & Yang, 2002).

No obstante, durante los últimos años se trabaja en la implementación de modelos y reglas utilizadas para mejorar el proceso de composición de servicios Web. Técnicas como WS-COORDINATION (Curbera, Khalaf, Mukhi, Tai, & Weerawarana, 2003) y WS-TRANSACTION (Ebbers, y otros, 2004) han hecho posible tener un manejo y control de la información transferida. Estas describen un marco extensible para proporcionar protocolos que coordinan las acciones de las aplicaciones distribuidas, utilizadas para apoyar una serie de aplicaciones, y definir una serie de condiciones las que coordinan el proceso de transferencia de mensajes. Con ellas, se pretende tener un control que garantice buenas comunicaciones entre los servicios. Por otro lado, se ha implementado una técnica que apoya el proceso de seguridad en la transferencia de mensajes WS-SECURITY (Nadalin, Kaler, Monzillo, & Hallam-Baker, 2006), ésta proporciona un lenguaje de seguridad para los servicios Web, suministrando intercambio de credenciales, integridad de mensajes y confidencialidad de los mismos, haciendo posible un mejor manejo de autentificación y cifrado (Berardi, 2004).

Otra de las técnicas que se utiliza para realizar procesos de composición es la planificación; planificar es el razonamiento sobre la acción. Este es un proceso de deliberación explícita y abstracta que escoge y organiza las acciones, anticipándose a los

resultados esperados (Ghallab, Nau, & Traverso, 2004). Esta deliberación, busca alcanzar de la mejor manera posible, algunos objetivos preestablecidos. La planificación automática, es un área de la Inteligencia Artificial (IA), que estudia este proceso de deliberación de manera computacional.

Un problema de planificación es visto como un proceso para encontrar una secuencia de acciones que partiendo de unas condiciones iniciales, alcanzan unas condiciones finales. Desde el punto de vista de agentes inteligentes, el campo de la planificación busca construir algoritmos de control que permitan a un agente sintetizar una secuencia de acciones que le lleve a alcanzar sus objetivos (Weld, 1999).

El problema de planificación se describe por la tupla $<S, s_o, G, A, \gamma>$ donde S es el conjunto de posibles estados, $s_o \in S$ denota el estado inicial de planificador, $G \subset S$ denota el conjunto de estados finales que el sistema de planificación alcanza, A, es el conjunto de acciones que el planificador ejecuta para pasar de un estado del mundo a otro y la transición $\gamma \subseteq SxAxS$ define la semántica de cada acción para describir el estado que resulta cuando una acción es ejecutada en un estado del mundo (Carman, Serafini, & Traverso, 2003).

1.3 Los riesgos en los servicios web

Uno de los problemas dentro de los ambientes dinámicos, es identificar y controlar aquellas variaciones que conducen a los sistemas a actuar de forma distinta a como lo venían haciendo. En la literatura se encontró que estas variaciones se enmarcan dentro del contexto del riesgo el cual, es analizado desde dos puntos de vista:

- Conceptual, donde se consideran como sucesos futuros que generan pérdidas o acontecimientos adversos que infieren en el cumplimiento de los objetivos pero que se evitan o mitigan.
- Estadístico, en el cual se sugiere como riesgo a un evento inestable haciendo referencia a la probabilidad que algo malo ocurre y la magnitud de su pérdida, desastre u otro suceso indeseable.

En la Web, las vulnerabilidades e inestabilidad de ésta, así como el diseño de los servicios Web y su integración con otros servicios, inducen a que éstos sufran alteraciones que afectan su comportamiento, rendimiento y manejo de la información ya que tales servicios están en constante funcionamiento y actúan de forma reactiva a cualquier cambio que se presente en su entorno. Por lo tanto, al manejar sistemas en función de servicios siempre estará la posibilidad que se presente un riesgo que le ocasione inestabilidad.

A continuación, se presenta la definición del riesgo y falla dentro del marco de este trabajo de tesis.

Definición 1.1: el riesgo, es uno o un conjunto de eventos enfocados a la mala suposición del mecanismo de composición sobre la información manejada por un servicio, la disponibilidad del servicio y la reactividad del servicio teniendo en cuenta restricciones de tiempo de ejecución; estos dos últimos analizados en periodos de tiempo (días, horas); los cuales alteran el comportamiento de un servicio causándole impactos negativos para

cumplir sus objetivos, y los cuales se miden a través de la asociación de los valores de cada evento.

Dentro de la composición de servicios, un riesgo implica que el servicio o el conjunto de servicios seleccionados sean utilizados de forma limitada ya que estos establecen un resultado contrario al esperado por un cliente. Es decir, el resultado de la composición tendrá como consecuencia una falla y por lo tanto, la falla de este servicio se utiliza en futuras composiciones.

Definición 1.2: una falla es un valor cuantificable producto de una inconsistencia del servicio Web o el sistema de composición como resultado de la evaluación de un riesgo.

Generalmente, cuando un riesgo es identificado, el experto humano es el encargado de realizar las correcciones necesarias para evitar eventos que afecten el comportamiento de los servicios. Sin embargo, al trabajar con composición de servicios de manera automática, se pretende que el experto humano no tenga participación dentro de este proceso ya que el experto sesga la información conforme a sus preferencias sin tener en cuenta que esta información está sujeta a cambios ocasionados por variaciones en su entorno que a diferencia de las máquinas, éstas, toman la información adicional oculta para el humano y obtener nuevos datos para realizar un mejor análisis. No obstante, esto no significa que el experto humano no requiera de la información del riesgo; al contrario, este tiene que conocer la información para actuar en momentos que sea requerido (Nam, Hyunyoung, & Lee, 2009).

Por lo tanto, cuando existe uno o más riesgos presentes en la composición de servicios, es necesario ejercer un control automático sobre estos riesgos y definir como estos riesgos, afectan a los servicios con el fin de determinar la participación de los servicios en el proceso de composición. El manejo de riesgos, proporciona los mecanismos para seleccionar aquellos riesgos y comprenderlos de acuerdo al objetivo de composición. De esta manera, se garantiza que el riesgo tenga un impacto acorde al ambiente de ejecución del servicio, esto es, afectar lo menos posible el comportamiento del servicio o servicios compuestos resultantes y así evitar fallos en dicha composición (Verdon & McGraw, 2004).

1.4 Composición de SW y el aprendizaje de máquinas

1.4.1 Aprendizaje de máquinas.

El aprendizaje de máquinas, es una rama de la IA, la cual pretende adquirir un conocimiento a partir de métodos programáticos. Este, se refiere a cambios en sistemas que utilizan IA como en el caso de predicciones, planificación, diagnósticos, reconocimiento de patrones entre otros y estos cambios se interpretan como mejoras (Dietterich, 2003). Para esto, toma la información del entorno y parte de éstas para emitir un juicio de manera que se cuente con una información para actuar en el futuro (Russell & Norving, 2004).

Un sistema de aprendizaje de máquinas considera las siguientes características (Mitchell, 1997):

- Selección de datos de entrenamiento: estos datos corresponden al conjunto de información obtenida de manera automática o manual por medio de un agente humano o un programa y a través de la cual, se infiere el conjunto de decisiones a seguir por el aprendizaje. Esta información, está directamente relacionada con el comportamiento del aprendizaje es decir, que sea determinante en el buen o mal funcionamiento del sistema.
- Función Objetivo: identifica cual es el tipo de conocimiento que será aprendido y como éste será utilizado en la ejecución del programa.
- Representación del Conocimiento: el sistema de aprendizaje requiere de un lenguaje para interpretar la función objetivo y el conjunto de datos de entrenamiento.
- Algoritmos de aprendizaje: definir cuál es el modelo algorítmico que se adapta a las características particulares que se desea aprender.

1.4.2 Técnicas de aprendizaje de máquinas

En la literatura se encuentran diferentes técnicas de aprendizaje de máquinas que se han utilizado a lo largo del tiempo para resolver diferentes problemas que se presentan en la vida cotidiana, sin embargo, en cuanto a la composición de servicios, las técnicas comúnmente utilizadas para adquirir la información en procesos de composición son: los árboles de decisión (Nam, Hyunyoung, & Lee, 2009), la programación lógica inductiva (Carman & Knoblock, Learning Semantic Descriptions of Web Information Sources, 2007), el razonamiento basado en casos (Liu, QIU, TAO, ZANG, & WANG, 2009), el aprendizaje por refuerzo (Wang & Tang, 2008) y el aprendizaje bayesiano (Li, Su, & Yang, 2007).

Arboles de Decisión (TDIDT). También conocidos como árboles de clasificación o Top Down Induction Descision Trees (TDIDT) (Qulnlan, 1986), (Araujo Sierra, 2006). Es un modelo de aprendizaje que se caracteriza por la sencillez de su modelo, fácil comprensión, su accesibilidad, la explicación que aporta y la rapidez al momento de clasificar nuevos patrones.

Esta técnica, se realiza utilizando procesos de inducción, su algoritmo consiste en hacer una clasificación de instancias para encontrar un pequeño árbol de decisión que ajuste un conjunto de datos (ejemplos). Este árbol, está compuesto por nodos de decisión internos y hojas terminales. Cada nodo especifica una prueba de algún atributo de la instancia, y cada rama del nodo corresponde a uno de los posibles valores del atributo. Las hojas del árbol son etiquetadas con un valor para el concepto objetivo que ajusta los ejemplos que satisfacen las condiciones a lo largo del camino de la raíz del árbol a las hojas. El proceso inicia en la raíz y se repite hasta llegar a un nodo hoja y el valor escrito en la hoja es el resultado.

El proceso consiste en tomar un conjunto de ejemplos entrenados y dividirlos acorde a un atributo seleccionado que minimice una medida de varianza a lo largo de la variable de predicción. Esta variable, es de dos tipos: discreta o continua, dando como resultado un árbol de clasificación o uno de regresión (Karalic, 1992) respectivamente.

Para saber cuál es la variable que será seleccionada y proceder con la división, es posible utilizar métodos heurísticos que aunque no evalúen todas las alternativas, consiguen resultados satisfactorios. Uno de los métodos heurísticos más utilizados es la entropía I; esta es una función diferencial utilizada para problemas de optimización y que

representa la impureza de los datos, esta es una medida de cómo está ordenado el universo (ver Ecuación (1.1)).

$$I(\mathrm{E}) = -\mathrm{p}\oplus \log 2\ \mathrm{p}\oplus - \mathrm{p}\ominus \log 2\ \mathrm{p}\ominus \quad (1.1)$$

Donde:
E: es el conjunto de ejemplos
p $\oplus$: es el conjunto de ejemplos positivos tomados del total de ejemplos S
p $\ominus$: es el conjunto de ejemplos negativos tomados del total de ejemplos S
p $\oplus$ + p $\ominus$= 1

La entropía también permite ser calculada al existir múltiples clases (ver Ecuación (1.2)).

$$I(E) = \sum_{i=1}^{C} -P_i\ log_2\ P_i \quad (1.2)$$

Donde:
P_i : es la proporción de E que pertenece a la clase i
C: son los valores posibles

Por otro lado, para medir la impureza del atributo I(A), se utiliza la siguiente ecuación (ver Ecuación (1.3))

$$I(A) = \sum_{i=1}^{n} \frac{|E_i|}{E}\ I(E) \quad (1.3)$$

Donde:
E: es el conjunto de ejemplos
n: posibles valores de un mismo atributo

A parte de medir la impureza de los datos, los árboles de decisión permiten la reducción de la entropía causada al particionar los ejemplos de acuerdo a un atributo *A*, esto lo realiza a través de la ganancia de información (ver Ecuación (1.4)).

$$gain(a) = I(E) - I(A) \quad (1.4)$$

Un problema que presenta esta técnica es la sobreajuste (overfitting), es decir, las hipótesis describen muy bien las instancias del aprendizaje, pero el error de la clasificación crece en ejemplos independientes de las pruebas. Para este problema, la solución es la utilización de métodos de poda; esto consiste en remover el subárbol que no tengan suficientes datos para la toma de decisiones confiables, haciéndolo un nodo hoja y asignándole la clasificación más común de los ejemplos entrenados asociados a ese nodo (Quinlan, 1993).

Una extensión a estos árboles son los árboles de decisión relacionales o lógicos (Blockeel & Raedt, 1998), los cuales extienden los árboles clásicos con lógica de primer orden. Una de las diferencias es trabajar con descripciones en un lenguaje racional como la lógica de predicados permitiendo hacer descripciones de hechos lógicos sobre cada ejemplo evaluado.

Programación Lógica Inductiva (ILP) (Muggleton, 1991), (Muggleton & Raedt, 1994), (Lavrac & Saso, 1994), (Kersting, 2006), (Araujo Sierra, 2006). Esta técnica, se refiere a una relación entre el aprendizaje inductivo y la programación lógica. Desarrolla descripciones de predicados a partir de ejemplos y conocimiento previo, y su representación se hace a través de programas lógicos.

El propósito de esta técnica es aprender una hipótesis que cubra los ejemplos positivos y no los negativos. Para ello, incluye técnicas como resolución inversa (Muggleton & Buntine, 1988), generalizaciones menos generales (Plotkin, 1969), implicación inversa (Muggleton, 1992) y parten de pruebas y modelos teóricos para calcular predicados de primer orden.

Esta técnica tiene varias características particulares, entre ellas su fácil comprensión por el humano ya que su resultado es expresado a través de inferencias lógicas, además soporta un conocimiento previo y tiene un lenguaje de refinamiento para mejorar la selección de hipótesis.

Las técnicas para aprender programas lógicos se enmarcan dentro de dos grupos: técnicas de generalización y las técnicas de especialización.

- *Técnicas de Generalización*

Una de las formas para estructurar el espacio de hipótesis, es a través de una $\theta - \text{Subsunción}$. Es decir, tratar de incluir una cláusula dentro de otra, por lo tanto si una cláusula C, $\theta - \text{subsume}$ (es una generalización de) una cláusula D si existe una subsunción θ tal que $C\theta \subseteq D$. Donde la subsunción es una serie finita de variables asociadas con términos.

Generalización Menos General (lgg): es una de las formas para encontrar hipótesis. La lgg de dos cláusulas C1 y C2, es la generalización más específica de las cláusulas C1 y C2 dentro de un ordenamiento parcial generado por una θ-subsunción.

Generalización Menos General Relativa (rlgg): utiliza la noción de lgg. Se basa en algoritmos de generalización los cuales utilizan la búsqueda de abajo hacia arriba (bottom-up) de una serie de θ – subsunción ordenados parcialmente. En ésta, se asume que si dos cláusulas C1 y C2 son verdaderas, es muy probable que lgg(C1,C2) sea verdadera.

Resolución Inversa: es una de las técnicas de generalización utilizadas para aprender programas lógicos. Se basa en el teorema de resolución de Robinson (Robinson, 1965). Esta es utilizada para generar hipótesis candidatas h que satisfagan $(B \wedge h_i \wedge x_i) \vdash f(x_i)$ donde B es el conocimiento previo (ver Ecuación (1.5)).

$$C_2 = (C - (C_1 - \{L_1\})\theta_1)\theta_2^{-1} \cup \{L_2\} \quad (1.5)$$

Donde:
C_1 y C_2: Son dos cláusulas
θ_1 y θ_2:Son substituciones que involucran las variables de las cláusalas C_1y C_2

- *Técnicas de especialización*

Búsqueda de arriba hacia abajo (Top Down) de grafos de refinamiento: los grafos de refinamiento son definidos y utilizados para guiar la búsqueda de la hipótesis más general a la específica. Los grafos son representados a través de nodos y arcos; los primeros representan las cláusulas del programa y son los operadores de refinamiento es decir, sustituyendo una variable por un término y adicionar un literal al cuerpo de la cláusula.

Razonamiento basado en casos (Aamodt & Plaza, 1994), (Kolodner, 1993). Es una técnica inspirada en el razonamiento humano y la organización de la memoria. Consiste de una memorización de problemas resueltos previamente (casos) a través de experiencias pasadas y propone una solución a problemas futuros similares.

Un caso se describe como la tupla $(\mathrm{P}, \mathrm{S}, \mathrm{E})$: problema, solución y efectos. Donde el problema es resuelto en eventos pasados, la solución es una descripción del camino del problema resuelto y los efectos son una descripción del resultado de aplicar la solución al problema.

El razonamiento basado en casos se organiza de tres formas posibles:

- Plano: es la forma más simple, permite realizar la eliminación y adición de casos de forma ágil, sin embargo, la búsqueda se realiza caso por caso y esto afecta la recuperación de los casos y el tiempo en el que se lleva este proceso.
- Cluster: parte de un modelo dinámico de memoria en el cual, los casos son almacenados dependiendo de la similaridad de los casos; esta similaridad se asocia de acuerdo a como las experiencias se utilizan o a la similaridad de los casos prototipos.
- Jerárquico: se utiliza cuando los casos comparten algunas de sus características y son agrupados. La estructura de memoria que se utiliza es similar a una red de categorías, relaciones semánticas, casos e índices. Los casos se asocian a una categoría y esta categoría es enlazada en una red semántica que contiene las características y los estados intermedios referentes a otros términos.

Usualmente, esta técnica sigue un ciclo de cuatro etapas las cuales son (Kolodner, 1983), (Kolodner, 1993):

- ✓ Recuperación: esta etapa después de evaluar el objetivo del problema, realiza una búsqueda del mejor caso dentro de una base de casos el cual, se aproxime a la solución del problema y por lo tanto una solución aproximada sea recuperada.

La recuperación se asocia a diferentes algoritmos de búsqueda que le permitan adquirir de su memoria, los casos más similares al problema. Sin embargo, entre los más comunes está el algoritmo del vecino más cercano, (ver Ecuación (1.6)), el cual busca la similaridad en todos los casos en la base de casos.

$$\frac{\sum_{i=1}^{n} w_i * sim\left(f_i^I, f_i^R\right)}{\sum_{i=1}^{n} w_i} \qquad (1.6)$$

Donde:
wi: es el peso de importancia de un atributo
sim: es la función de similitud
f_i^I y f_i^R: son los valores del atributo i en el caso de entrada (I) y el caso recuperado (R)

Otro algoritmo es asociar ranking a los casos, es decir asociar valores estadísticos a los casos en la base de casos.

Otro tipo de búsqueda es la búsqueda en paralelo, la cual es posible cuando el emparejamiento del caso no requiere intercambio de mucha información entre los procesos de ejecución en paralelo. Este algoritmo, se utiliza cuando los casos están organizados de forma plana o en cluster.

- ✓ Adaptación: mapea la solución recuperada y la adapta para ajustarla de mejor manera al nuevo problema.

Esta es de dos tipos: estructural o derivada. La primera aplica reglas directamente a la solución almacenada en los casos. Esta adaptación, incluye modificaciones a parámetros. El segundo reutiliza reglas, métodos, algoritmos que genera la solución original para producir una nueva solución al problema actual.

- ✓ Revisión: evalúa la solución adaptada antes o después de que la solución sea aplicada al problema recuperado. Si la solución no es satisfactoria, se adapta nuevamente o se buscan más casos.
- ✓ Actualización: si la solución se adapta al objetivo del problema, el nuevo caso es almacenado en la base de casos y está disponible para nuevos usos.

Aprendizaje por Refuerzo (Kaelbling, Littman, & Moore, 1996): es una técnica que aprende cual es la mejor acción para alcanzar un determinado objetivo a través del ensayo y error en un ambiente dinámico y su recompensa asociada.

Generalmente, el aprendizaje por refuerzo trata de encontrar una regla de decisión (política π) que representa la distribución de probabilidad asociada a una acción, la cual le permite decidir cuál es la mejor acción a seguir en un determinado momento. Generalmente esta técnica de aprendizaje utiliza el marco de los procesos de decisión de Markov (MDPs) y para esto, tiene en cuenta los siguientes aspectos:

- ✓ Un conjunto de estados S.
- ✓ Un conjunto de acciones A.
- ✓ La transición de estado T.
- ✓ Recompensa o refuerzo. R.

El proceso consiste en evaluar la interacción del ambiente y el agente que lo monitorea de tal manera que se perciba el estado actual S en el que se encuentra y se decide ejecutar una acción A como salida. En respuesta, el ambiente retorna una transacción estocástica a un nuevo estado S' y estocásticamente se emite una recompensa numérica R. Finalmente el agente busca maximizar la recompensa que adquiere a lo largo de las ejecuciones.

Varios algoritmos de aprendizaje por refuerzo tratan de aprender la política π utilizando aproximaciones de funciones de valor las cuales, determinan que tan bueno es ejecutar una acción en un determinado estado. Sin embargo, en ocasiones no se tiene conocimiento de la probabilidad de la transición del estado y la función de refuerzo (Modelo) y por lo tanto el agente tiene que interactuar con el ambiente con el fin de obtener dicha información. Para ello, existen dos caminos:

- Aprendizaje por refuerzo basado en el modelo (Strehl, Li, & Littman, 2006)

Una técnica que implementa este camino de soluciones es la iteración de valor (Busoniu, Babuska, Schutter, & Ernst, 2010). Este algoritmo se basa en las actualizaciones del valor de los estados. Como valor inicial, la utilidad del estado es asignada por el valor de una función de recompensa en el estado $V'(s)$. (ver Ecuación (1.7)).

$$V'(s) = \max\left(\sum_{s'} P_a\ (s, s') * \left(R_a(s, s')\right) + \gamma\ V(s')\right) \quad (1.7)$$

Donde:
S y S': son los estados actual y nuevo estado respectivamente
P_a: es la función valor acción
R_a: es la función de refuerzo
V: función de valor-estado
γ: factor de descuento para futuras acciones

Otra técnica es la iteración de política (Gosavi, 2004). Este algoritmo se basa en actualizaciones de una política π la cual escoge la acción con respecto a la función de utilidad. (ver Ecuación (1.8)).

$$V'(s) = \sum_{s'} P_{\pi(s)}\ (s, s') *\ V(s')) \quad (1.8)$$

Donde:
S y S': son los estados actual y nuevo estado respectivamente
P: es la función valor acción
V: función de valor-estado

- Aprendizaje por refuerzo Libre del modelo (Strehl, Li, Wiewiora, Langford, & Littman, 2006)

A diferencia de los métodos basados en modelos, éste solo se basa en la interacción con el entorno. Entre los principales métodos utilizados está el método de Monte Carlo (Barto & Duff, 1994) y los métodos de diferencia temporal (DT) (Watkins, 1989).

El método de Monte Carlo, se basa en la actualización de las funciones de valor mediante secuencias de resolución del problema. Dado que el modelo no es conocido, se requiere aprender la función Q de valor-acción en lugar de la función valor-estado. Este algoritmo, utiliza una lista de ganancias para almacenar las ganancias obtenidas al ejecutar la acción a, desde el estado s y se calcula el valor promedio de esa lista para actualizar la función Q. (ver Ecuación (1.9)).

$$\pi = argMax\, Q(s,a) \tag{1.9}$$

Método de diferencia temporal (DT): estos realizan estimaciones en función de otras estimaciones adquiridas anteriormente. Uno de los algoritmos más utilizados es el Q-Learning: este algoritmo aprende a partir de la exploración del entorno. En este algoritmo, se adiciona un parámetro más que representa la diferencia entre los valores de las nuevas actualizaciones y los valores adquiridos anteriormente, sin embargo, este algoritmo solo se utiliza cuando el dominio es estocástico (ver Ecuación (1.10)).

$$Q(s,a) \leftarrow r + \gamma \max Q\,(s',a') \tag{1.10}$$

Donde:
s, s':son el estado de ejecución actual y el nuevo estado luego de ejecutar la acción a respectivamente
a, a': acciones en el estado s y s'respectivamente
Q: es la función de utilidad estimada
$Q(s,a)$: es la recompensa inmediata para realizar una acción
r: recompensa inmediata
γ:valor relativo del retraso del refuerzo vs la recompensa inmediata

Aprendizaje Bayesiano (Wellman, Breese, & Goldman, 1992), (Poole, 1993), (Sebe, Cohen, Garg, & H., 2005): esta técnica modela un fenómeno mediante un conjunto de variables y las relaciones de dependencia entre ellas. Ésta, permite estimar la probabilidad posterior de las variables no conocidas. Gráficamente su representación está dada a través de nodos los cuales describen las variables aleatorias y los arcos las relaciones de dependencia directa entre las variables.

Este aprendizaje busca encontrar la hipótesis más probable dado un conjunto de datos de entrenamiento y un conocimiento a priori de la probabilidad de esa hipótesis. Para ello, esta técnica se basa en el teorema de bayes (Swinburne, 2005) (ver Ecuación (1.11).

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{1.11}$$

Donde:
$P(h)$: es la probabilidad a priori de la hipótesis h.
$P(D)$: es la probabilidad de que se del conjunto de entrenamiento D.
$P(D|h)$: es la probabilidad de observar el conjunto de entrenamiento D dada la hipótesis h.
$P(h|D)$: es la probabilidad a posteriori de h, una vez observado el conjunto de entrenamiento D.

Una de las formas para calcular la hipótesis más probable (hMAP) (Abramovich & Angelini, 2006) , es a través del teorema de bayes (ver Ecuación (2.12)).

$$\mathrm{hMAP} = \mathrm{argmax}_{h \in H}\,(P(h \mid D))$$

$$= \mathrm{argmax}_{h \in H}\left(\frac{P(D|h)P(h)}{P(D)}\right)$$

$$= \operatorname{argmax}_{h \in H} (P(D \mid h) P(h)) \quad (1.12)$$

Donde:
$P(h)$: es la probabilidad a priori de la hipótesis h.
$P(D)$: es la probabilidad de que se del conjunto de entrenamiento D.
$P(D|h)$: es la probabilidad de observar el conjunto de entrenamiento D dada la hipótesis h.
H: es el conjunto de hipótesis

De otro lado, si las probabilidades de las hipótesis son igualmente probables, se calcula a través de la probabilidad de los datos y cualquier hipótesis que maximice se llama máxima probabilidad (Lee & Song, 2003), (ver Ecuación (1.13)).

$$h_{ML} = \operatorname{argmax}_{h \in H} (P(D \mid h)) \quad (1.13)$$

Donde:
$P(D|h)$: es la probabilidad de observar el conjunto de entrenamiento D dada la hipótesis h.

1.5 Aproximaciones relacionadas con la composición de sw y el manejo de riesgos de falla

1.5.1 Puntos de comparación

En el marco de esta tesis, se han definido los siguientes puntos de vista de comparación para analizar las aproximaciones relacionadas con la composición de servicios y el manejo de riesgos de fallas.

- Manejo de riesgos. Este, determina si el trabajo contempla el manejo de riesgos durante el proceso de composición de servicios, sus posibles valores son: Sí y No.
- Técnicas de asociación. Pretende identificar el tipo de técnica que los trabajos de investigación utilizan para adquirir la información del riesgo. Sus valores posibles son: Experto, Automático o Asociación de terceros.
- Nivel de análisis de riesgos. Este criterio especifica en que parte se realiza el análisis de fallas, sus posibles valores son: riesgos ocasionados en el entorno de los servicios y riesgos ocasionados en el entorno del compositor de servicios.

- Aspectos de calidad. Determina si el trabajo contempla el uso de criterios QoS en la obtención de la información que evalúa el comportamiento de los servicios. Sus posibles valores son: disponibilidad, confiabilidad, temporalidad, reactividad, costo computacional, costo de transmisión, reputación, Atomicidad, consistencia, aislamiento, durabilidad, ejecuciones exitosas.

1.5.2 Aproximaciones

El trabajo de (Cardoso, Sheth, Miller, Arnold, & Kochut, 2004), se enfoca en el manejo de QoS para mejorar la composición de servicios, en el que se describe un modelo que permite predecir la calidad de los servicios participantes en flujos de trabajo. Aclara que

se compensa la deficiencia de la composición, si existen muchos servicios que tengan funciones similares. En este trabajo, se identifican características cuantitativas de los servicios es decir, aquellas medidas que se evalúan de manera concreta. Estas características son: el costo de ejecución, el tiempo de respuesta y la confiabilidad siendo esta última la característica que mide el riesgo de falla. Para esto, la confiabilidad evalúa las probabilidades de fallos del sistema y el número de fallos de proceso. Los fallos del sistema, son errores de software; mientras que los fallos de proceso, es la relación entre las veces que el servicio no se ejecutó y el número de llamados del servicio para ser ejecutado. El modelo de predicción está enfocado a un modelo de distribución de probabilidades que le permite adquirir el valor de probabilidad asociado a cada uno de los criterios. Los valores de los criterios, en primera instancia son asignados por un experto y a medida que el servicio sea ejecutado, se calcula las probabilidades de cada criterio. El manejo de riesgos en este trabajo se complementaría con información adicional como datos periódicos de ejecuciones y análisis de los servicios para determinar el tiempo que estos requieren para ejecutarse y especifique mejor el comportamiento del servicio. Por otro lado, en este trabajo se identifica la valoración de las fallas a nivel de los servicios, dejando de lado la valoración del compositor de servicios el cual también presenta riesgos de falla que impiden el buen comportamiento del proceso de ejecución.

En (Liu, Zhu, Li, Li, & Cui, 2009) utiliza las QoS como información adicional para mejorar las composiciones de servicios. Propone un modelo de servicio que evalúa los QoS a través de la asignación de pesos a los atributos a través de tres fuentes: el proveedor del servicio, la experiencia de los usuarios y el monitoreo del servicio. Cada peso y atributo, son sumados de manera que se defina el valor de calidad del servicio. En esta aplicación, el control que se ejerce sobre la asignación de los pesos, está asociado a la utilización del servicio de manera que si el servicio se solicita de manera activa, éste tendrá un peso más alto en el valor de retroalimentación; mientras que si el servicio no es utilizado obtiene el peso asignado por el proveedor del servicio. En este trabajo, la retroalimentación del servicio a través de las ejecuciones del servicio determina el comportamiento de éste, sin embargo no valida el comportamiento del los servicios bajo diferentes circunstancias de tiempo lo que mejoraría la retroalimentación. Este trabajo se concentra en la valoración de los servicios, dejando de lado la evaluación del compositor.

El manejo de riesgos es analizado en (Liu, Zhang, Ren, Liu, & Zhang, 2009), donde se muestra un enfoque para reducir las fallas relacionadas con el costo computacional y el costo de comunicación causadas por problemas en la ejecución de servicios durante su composición. Cada falla es calculada a través del impacto del riesgo de falla. Para ello, analiza por un lado la dependencia de fallo entre los servicios, es decir, identifica si un servicio depende de otro, para lo cual el servicio dependiente se cancela o compensa. Por otra parte, analiza la dependencia de compensación es decir, si un servicio depende de otro, este servicio será compensado. Asimismo, la evaluación del riesgo de falla por pérdida para cada servicio, es evaluado a través de dos funciones: la primera evalúa el costo de ejecución y el costo del servicio y la segunda, evalúa el tiempo de ejecución. Cada una de estas funciones tiene asignado un valor de importancia, el cual es asignado por el usuario. Aunque se define claramente los tipos de falla que son manejados, la información de la falla, se obtiene a través de una probabilidad de éxito del servicio sin importar su entorno, es decir se evalúa el riesgo de falla que existe sin evaluar en qué momento o bajo qué circunstancias se presenta, esto daría más precisión en la asignación del valor para el riesgo. Además, el trabajo se enfoca en la evaluación del riesgo de falla en los servicios valorando únicamente su entorno y deja de lado las fallas ocasionadas por el compositor de servicios.

Por otro lado, (Kokash & D'Andrea, 2007), (Kokash, 2008) plantea un enfoque para la selección de servicios en el cual, se trata de minimizar el impacto de fallas de servicios web atómicos por medio de sus QoS. Este trabajo, busca garantizar al usuario confiabilidad en los resultados. Para ello, evalúa los componentes y datos del servicio, la probabilidad que se dé un evento negativo, defectos en el diseño del software y el impacto que éstos tendrían en cuanto a términos monetarios y de reputación del servicio. El riesgo es medido a través de la probabilidad de falla del servicio y el resultado de su impacto. La probabilidad de falla se mide a través de las ejecuciones fallidas del servicio atómico y el resultado del impacto se caracteriza por utilizar una función de pérdida en la que se evalúa el costo de ejecución del servicio y el tiempo de transferencia del mensaje. En general este trabajo pretende minimizar el riesgo de un plan de composición a través de la minimización de los riesgos de cada servicio del plan.

Las variaciones de ejecución llegan a ser determinantes para un análisis de riesgos. En este trabajo, la ejecución determina el impacto del riesgo pero no considera la variación de las ejecuciones en periodos de tiempo representados en horas y días, dando la posibilidad a su sistema de tomar una decisión equívoca cuando un servicio es solicitado en determinado instante en el tiempo. Además, solo se considera el impacto del riesgo del servicio y no sobre su composición, dejando de lado datos relevantes que mejoran la identificación de las fallas y contribuiría a garantizar a un más los resultados esperados por los usuarios.

El trabajo de (El Haddad, Manouvrier, Ramirez, & Rukoz, 2008) se orienta a la selección y composición de servicios apoyado en un modelo de calidad el cual funciona de acuerdo a la asignación de pesos a través de preferencias del usuario. Se plantean dos tipos de riesgos durante la ejecución: el primero, verifica que la ejecución del servicio sea exitosa y de ser esto cierto, el usuario califica a ese servicio con un peso de compensación por su buen funcionamiento. El segundo riesgo, mide lo contrario es decir, que el servicio no se ejecuta.

Los riesgos medidos en este trabajo son expectativas que tiene el usuario de un buen comportamiento en ejecución del servicio, sin embargo, esto pude ser ambiguo de manera que el comportamiento de los servicios está relacionado con el ambiente en el cual se evalúa y no depende del usuario. La valoración del ambiente de ejecución de manera automática, obtendría mejores resultados porque éste es un monitoreo constante del servicio, capturando información desconocida para el usuario como lo es el comportamiento de los servicios en diferentes circunstancias de tiempo. Además, estos datos obtienen información importante de los riesgos del compositor y no limitarlos solo a la valoración del entorno del servicio.

El trabajo de (Cardinale, El Haddad, Manouvrier, & Rukoz, 2010), presenta un algoritmo para mejorar la selección de servicios en una composición. Este algoritmo, se basa en redes de petri el cual, evalúa las entradas y salidas del servicio, sus propiedades transaccionales es decir, la atomicidad, consistencia, aislamiento y durabilidad, y evalúa el nivel de riesgo. Este riesgo, es un valor asociado a los servicios por parte de un usuario a las propiedades transaccionales de manera que la suma de éstas, determina el servicio a ser seleccionado en el proceso de composición. A diferencia de los trabajos mencionados anteriormente, este trabajo no hace uso de QoS para adquirir la información de riesgo. En este trabajo el riesgo es interpretado como un valor agregado

del usuario, el cual define según su criterio, que valor asignar a cada propiedad. No obstante como se ha mencionado, el criterio del usuario llega a ser subjetivo si no cuenta con bases fundamentadas. A diferencia de éste, el monitoreo de las ejecuciones de los servicios de forma automática, permitiría adquirir información tanto del entorno del los servicios para identificar su comportamiento, como el entorno del compositor para identificar anomalías que se presentan en éste.

En el trabajo de (Chan & Bishop, The Design of a self-Healing Composition Cycle for Web Services, 2009), se discute el uso cinco requerimientos para apoyar la composición de servicios utilizando redes de tareas jerárquicas. Los requerimientos son: monitoreo de ejecución, análisis de fallas, sincronización, planificación y ejecución del plan. De estos cinco pasos, el análisis de fallos es el encargado de detectar las fallas, para ello se apoya en una taxonomía de fallas (Chan, Bishop, Steyn, Baresi, & Guinea, 2009) la cual tiene categorizada tres fallas importantes: fallas físicas, fallas de desarrollo y fallas de interacción entre servicios. A partir de este análisis se trata de omitir aquellos servicios que se consideran presentan una falla y no hacen parte del proceso de composición. Por lo anterior, la monitorización analiza solo riesgos producidos por los servicios. Por otro lado, la parte de análisis de fallas no está implementada de manera automática sino a través de la asignación de un valor por parte del usuario, esto hace que los datos de monitorización de los servicios no se actualicen con el paso de ejecución y la información se perdería y se tendría que realizar nuevamente el proceso de captura de información, lo que ocasiona demora en la construcción del plan de composición y que los servicios seleccionados no sean los más apropiados puesto que se tendría en cuenta solo la información adquirida actualmente, despreciando la información que ya se había obtenido y con la que se realizaría un mejor análisis de fallos porque se cuenta con información en diferentes ambientes de ejecución.

1.5.3 Comparación de aproximaciones

En los trabajos analizados, se identifica riesgos de falla presentados dentro del entorno de los servicios, los cuales son medidos a través del uso de diferentes QoS tales como costos de ejecución, número de ejecuciones exitosas, confiabilidad entre otras. Esto permite señalar que la información del riesgo de falla se obtiene únicamente del comportamiento de los servicios lo que implica que de este comportamiento depende su selección dentro de un plan de composición. Esta característica es eficiente, sin embargo, se descuida el estudio del compositor ya que éste al igual que los servicios presenta fallas en cuanto a las malas suposiciones de la información manejada por un servicio al instante de ser seleccionado como parte de un plan de composición.

El principal criterio que es evaluado en cuanto a los riesgos de falla es la probabilidad de éxito de los servicios, pero ésta se estudió sin tener en cuenta aspectos como periodos de tiempo o restricciones del tiempo en un periodo determinado lo que ayudaría a tener un conjunto de información más precisa sobre el comportamiento de los servicios durante su ejecución y mejora la expresividad de los riegos de falla. Por otro lado, algunos de los criterios que evalúan los riesgos de falla, son más determinantes cuando se evalúan en función del plan de composición al cual pertenecen. No siempre los valores individuales garantizan que el plan de composición sea el mejor.

En varios de estos trabajos prevalece el uso de datos asociados por un experto para determinar los valores del riesgo de falla para los servicios. Esto es bueno siempre y

cuando el número de servicios por analizar sea bajo ya que esto retrasa el proceso de composición porque el experto analizaría el valor de riesgo que es requerido por cada servicio. Por lo tanto, sería importante automatizar este proceso para hacerlo más eficiente.

Por otro lado, los valores de riesgo que el experto asigna a los servicios, se mejoran con información adicional que a diferencia de él, la monitorización de los servicios a través de QoS obtienen y hacen más precisa la evaluación del riesgo de falla.

La tabla 1.1 presenta la comparación de las aproximaciones para manejo de riesgos de falla en la composición de servicios.

Tabla 1.1: Comparación de las aproximaciones

Requerimiento / Trabajo	Técnicas de Asociación	Nivel de Análisis de Riesgos	Aspectos de Calidad
(Cardoso, Sheth, Miller, Arnold, & Kochut, 2004)	Experto, Automático	Entorno del servicio	Confiabilidad, Reputación Costo ejecución
(Liu, Zhu, Li, Li, & Cui, 2009)	Experto, Automático, Asociación de terceros	Entorno del servicio	Reputación, Tiempo de respuesta, disponibilidad
(Liu, Zhang, Ren, Liu, & Zhang, 2009)	Automático	Entorno del servicio	Costo computacional Costo de transmisión
(Kokash & D'Andrea, 2007), (Kokash, 2008)	Experto, Automático	Entorno del servicio	Costo de ejecución Ejecución exitosa
(El Haddad, Manouvrier, Ramirez, & Rukoz, 2008)	Experto	Entorno del servicio	Ejecución exitosa
(Cardinale, El Haddad, Manouvrier, & Rukoz, 2010)	Experto	Entorno del servicio	Atomicidad, consistencia, aislamiento y durabilidad,
(Chan & Bishop, The Design of a self-Healing Composition Cycle for Web Services, 2009)	Automático	Entorno del servicio	Ejecución exitosa

Particularmente en la tabla 1.1 se observa que los trabajos analizados en el estado el arte no hacen uso de periodos de tiempo para determinar los comportamientos dinámicos de los servicios en la Web, basándose únicamente en una evaluación global del servicio en el tiempo. Asimismo, se observa la constante de estos trabajos en solo enfocarse en la valoración del entorno de los servicios descuidando el comportamiento del compositor de servicios el cual también presenta fallos en la composición.

Por otro lado, la valoración del riesgo es asignado por parte de un experto, es decir deja la libertad de asignar el valor que tiene el criterio de cada servicio y además determina que criterio es más relevante que los otros para calcular el riesgo total de los servicios. Esto implica la subjetividad en el resultado final del riesgo ya que éste alcanza múltiples valores de acuerdo al usuario, ocasionando una variación de resultados finales y no tener certeza de cuál de estos es el que más se aproxima a la realidad del comportamiento del servicio.

1.6 Discusión

En este capítulo se realizó una revisión sobre los principales trabajos que evalúan el manejo de riesgos de falla en composiciones de servicios en los cuales se identificó, que para obtener la información para la evaluación del riesgo se hace uso de los QoS de los servicios.

Esta revisión permitió identificar algunos de los riesgos presentan los servicios Web durante el proceso de composición. Además, hizo posible la valoración de los trabajos existentes en los cuales se pudo encontrar la presencia de limitaciones que no fueron evaluadas y que implica un estudio adicional para ser tratadas. Éstas, están relacionadas con el manejo de periodos de tiempo para la valoración de cada criterio que evalúa el comportamiento de los servicios y por otro lado, el análisis del comportamiento del compositor de manera que se identifiquen las fallas relacionadas con éste.

Igualmente, se observó que en los trabajos valorados en la literatura, existe gran apoyo de los usuarios para asignar los valores de cada criterio que evalúan el comportamiento de los servicios y además, estos también asignan el nivel de importancia que tiene cada criterio respecto al otro. En otros casos, los cálculos solo se implementan con los datos que cada proveedor de los servicios les asigna, lo cual genera problemas porque el análisis que se realiza se hace sin evaluar el comportamiento dinámico de los servicios y la Web obteniendo solo un valor constante para todos los casos en los que se evalué un servicio.

Por lo anterior, esta tesis, se enfoca en el desarrollo de un modelo basado en aprendizaje de máquinas, que haga posible adquirir de forma automática el conocimiento relacionado con el factor de riesgo que presentan los servicios que participan en el proceso de composición, de manera que sea posible minimizar el riesgo de falla en una composición de servicios.

Con este estudio, se pretende demostrar la importancia de la asignación automática del valor para cada criterio que mide el riesgo de falla, teniendo en cuenta el entorno de los servicios, el entorno del compositor y la manejabilidad de varios riesgos en la valoración de riesgos de falla en una composición.

2. Definición de criterios para el manejo de riesgo de falla

Este capítulo, presenta los criterios utilizados para medir el factor de riesgo de falla en la composición de los servicios Web y las métricas asociadas a cada uno de ellos, así como el factor de riesgo calculado a través de esas métricas.

2.1 Criterios y métricas para el manejo de riesgos de falla

En las investigaciones encontradas (Cardoso, Sheth, Miller, Arnold, & Kochut, 2004), (Kokash & D'Andrea, 2007), (El Haddad, Manouvrier, Ramirez, & Rukoz, 2008), (Liu, Zhang, Ren, Liu, & Zhang, 2009), se encontró diferencias en cuanto a los riesgo de falla y la forma como estos se evalúan. Generalmente, el manejo del riesgo de falla está directamente relacionado a la asignación de un valor particular ya sea por parte de un experto o asignado por el proveedor del servicio, los cuales definen un conjunto de características que describen la información particular de los servicios para representar su comportamiento y medir el impacto que tiene el servicio al ser utilizado, definiendo una solución a partir de dicha medición.

Durante la revisión del estado del arte, se pudo identificar la utilización de QoS como una fuente para obtener la información que permite identificar el factor de riesgo de falla en composiciones de servicios. Algunos QoS, son definidos por la International Business Machines (IBM), los cuales, adhieren a los servicios una representación de calidad para mejorar su selección en la composición (Mani & Nagarajan, 2002). Estas características, han servido como base para que algunas investigaciones a través del tiempo las vayan adaptando e implementando en distintos modelos para tratar los riesgos. No obstante, también se pudo constatar la ausencia de indicadores de tiempo para cada QoS y a través de los cuales se especifica mejor el comportamiento de los servicios mediante un análisis más detallado donde se recurra a una evaluación de días y horas de la semana.
Los QoS, actúan principalmente sobre la ejecución de los servicios, obteniendo información de su comportamiento como: el número de veces que el servicio se ejecutó, el número de veces que no lo fue, el costo monetario que tiene su ejecución entre otros, y a partir de estos datos, calculan el riesgo de falla que tiene un servicio.

Definición 2.1: Un criterio, es definido como una característica de QoS de los servicios Web, a través de la cual, se pretende medir y evaluar una condición de riesgo al utilizar un servicio Web en la composición. Cada criterio, lleva asociada una expresión matemática que obtiene el valor medible del riesgo.

Generalmente, los trabajos presentados en el estado del arte, se concentran en un tipo específico de riesgo de falla que es el número de éxitos que tiene un servicio al ser ejecutado es decir, su probabilidad de éxito. Este riego de falla, se calcula desde diferentes aspectos como por ejemplo: el número de ejecuciones del servicio (Cardoso, Sheth, Miller, Arnold, & Kochut, 2004) y el costo monetario de ejecución (Kokash & D'Andrea, 2007) que implica, valores de castigo o compensaciones asignados por el usuario de acuerdo al comportamiento del servicio, o tiempos de transferencia de datos y ejecución (Liu, Zhang, Ren, Liu, & Zhang, 2009), (Kokash, 2008) entre otros. Sin embargo, este riesgo también es valorado desde el punto de vista temporal, es decir, identificar en qué periodo de tiempo el servicio tiene mejor comportamiento, en que día o a qué hora es más factible utilizar el servicio en un proceso de composición. La temporalidad permite un análisis a más detalle del comportamiento de los servicios en el que se pasa de tener un conocimiento general de su comportamiento a un conocimiento particular definido según el periodo de tiempo en el que es ejecutado el servicio.

Teniendo en cuenta lo anterior, el primer criterio seleccionado para evaluar el riesgo de falla de los servicios es la disponibilidad del servicio en periodos de tiempo.

Asimismo, un servicio se ejecuta normalmente y emite una respuesta en el tiempo que sea necesario, no obstante, al trabajar con restricciones en el tiempo de ejecución, se limita al compositor a seleccionar aquellos servicios que cumplen con dicha restricción. Esta característica también se asocia a periodos de tiempo de manera que se seleccione aquellos servicios que mejor se adapten a los requerimientos de restricciones de tiempo. Por lo tanto, el segundo criterio es la reactividad de los servicios en periodos de tiempo, haciendo énfasis al tiempo de restricción para ejecutar un servicio y alcanzar la respuesta al cliente.

Finalmente, hasta el momento solo se ha considerado el riesgo de falla en el entorno del servicio, No obstante, el riesgo de falla también está presente en el compositor de servicios. Una de las fallas presentes en éste, es el mal manejo de las creencias que un servicio requiere para ser ejecutado. Estas creencias representan el conjunto de datos de entrada y salida del servicio que el compositor asume serán las correctas después de la ejecución de servicios. Para evaluar este riesgo, el tercer criterio a evaluar en esta tesis será el manejo de creencias el cual permite hacer un seguimiento al compositor a través de la ejecución de los servicios y sus valores de la información asociada a los servicios que el supone estos manejarán.

De acuerdo a lo anterior, los riesgos de falla que serán evaluados en esta tesis son aquellos que involucran el comportamiento de los servicios y los que involucran el comportamiento del mecanismo de composición; siendo éstos, riesgos potenciales que tienen una ocurrencia constante y por lo tanto necesitan un seguimiento continuo, los cuales no se tratan en la literatura investigada. Cada uno de estos riesgos, tiene la posibilidad de ser identificado, cuantificado y evaluado para determinar cómo afecta la composición.

En consecuencia, los criterios que evaluarán estos riesgos son:

- Disponibilidad de los servicios en periodos de tiempo.
- Reactividad del servicio con restricciones de tiempo de ejecución asociadas a periodos de tiempo.

- Manejo de Creencias por el compositor de servicios sobre la información de entradas y salidas manejadas por cada servicio.

Los dos primeros criterios, evalúan el riesgo de falla del entorno del servicio, mientras que el último criterio evalúa un aspecto importante en el comportamiento del compositor como son las creencias que el maneja.

Descripción y Definición de los Criterios QoS y Métricas.

A continuación se presenta las descripciones de los criterios QoS, donde las definiciones y ecuaciones asociadas a estos criterios y a los riesgos, son propuestas en el marco de esta tesis.

- Criterio 1. Disponibilidad

La disponibilidad, es uno de los criterios que se estudia en varias investigaciones para obtener información sobre el comportamiento de los servicios Web, su forma de cálculo, se valora desde diferentes puntos de vista, en primer lugar, la disponibilidad se evalúa a través del número de invocaciones exitosas que hace un cliente a un servicio Web (Mikic-Rakic, Malek, & Medvidovic, 2005), (Nurm, Brevik, & Wolski, 2005).

Estas invocaciones exitosas, representan la probabilidad que tiene un servicio para ser utilizado. Para esta apreciación, se analiza el número de veces que un cliente invoca al servicio y el número de veces que el cliente no falla al acceder al servicio. Así, entre más altos sean los valores de probabilidad, la disponibilidad del servicio es muy alta. Si los valores son muy bajos, indican la imprevisibilidad del servicio para ser utilizado.

Otro enfoque para evaluar la disponibilidad de un servicio es aquel en el que ésta es vista como un periodo de tiempo en el cual se evalúa la cantidad de tiempo que un servicio es utilizado (Zeng, Benatallah, Dumas, Kalagnanam, & Sheng, 2003).

Finalmente, otra opción de medir la disponibilidad es a través de periodos de actividad e inactividad de los servicios, esto hace referencia a la frecuencia de interacción independiente de estos periodos durante un minuto de tiempo (Rosenberg, Platzer, & Dus, 2006).

Sin embargo, la disponibilidad de un servicio, también se mide teniendo en cuenta la frecuencia de invocaciones exitosas del servicio durante periodos de tiempo.

Definición 2.2: la disponibilidad de un servicio Web, es definida como la relación del número de éxitos del servicio Web en atender una petición y el número de veces que ese servicio se solicitó para atender dicha petición en el periodo de tiempo que es ejecutado. El tiempo es discriminado por el día de la semana y la hora en la que se hizo la petición.

La disponibilidad $D(s_i)$, evalúa el instante en el que el servicio Web está listo para ser utilizado, es decir se evalúa la frecuencia de las ejecuciones positivas en las cuales el servicio se encontraba visible para el cliente, teniendo en cuenta su participación en el plan de composición. La ejecución del servicio implica identificar el instante del tiempo valorado en días y horas para mejorar la expresividad de su rendimiento (ver Ecuación (2.1)).

$$D(s_i) = Frec(s_{i\ exitos}) \epsilon P \mid Ejecución(s_i) \rightarrow d, h \quad (2.1)$$

Donde:
$\mathrm{Frec}(s_{\mathrm{i\ exitos}})$: es la frecuencia de la invoación exitosa del servicio s_i.
P: es el plan de composición: $\sum s_i, \mathrm{goal}$; donde goal es el objetivo de composición.
d: es el dia que se llamó a ejecución al servicio s_i
h:es la hora en la que se ejecutó el servicio s_i

- Criterio 2. Reactividad de servicios

La reactividad en los servicios evalúa el tiempo de procesamiento de una solicitud, teniendo en cuenta su recepción, procesamiento y respuesta (Pereira, Franco, Silva, Meira, & Santos, 2004).

En algunos casos, se ha utilizado medidas como el tiempo de respuesta, el cual calcula el máximo, mínimo o promedio de tiempo requerido para completar los requerimientos (Gunther, 2000). Asimismo se apoya en otras características como la latencia y el procesamiento de información. La primera, representa el tiempo que existe entre el envió de una petición al servicio y el tiempo que tarda éste en revisarla; la segunda, evalúa el número de peticiones resueltas por un servicio en un periodo de tiempo (Ran, 2003).

Bajo condiciones de restricción de tiempo, algunos compositores prefieren seleccionar aquellos servicios que requieren menor tiempo de ejecución para formar parte de un plan de composición y construir un plan bajo mínimas condiciones de tiempo.

Definición 2.3: la reactividad de servicios, determina que tan bueno es utilizar un servicio Web bajo restricciones de tiempo de ejecución, teniendo en cuenta los periodos de tiempo discriminados por días y horas durante la semana en los que el servicio se ejecute.

La reactividad, calcula la frecuencia de los éxitos que tiene un servicio para responder a un cliente en un tiempo determinado, este tiempo es igual o menor al tiempo total de restricción emitido por el cliente (ver Ecuación (2.2)).

$$R(s_i) = \{ Frec(s_{i\ exitos}) \epsilon P\} t \leq T \mid Ejecución(s_i) \rightarrow d, h \ \ y \ \ t \in T = \{1,2 \dots n\ seg\} \quad (2.2)$$

Donde:
$\mathrm{Frec}(s_{\mathrm{i\ exitos}})$: es la frecuencia de los servicios exitosos pertenecientes a un plan
P:es el plan de composición: $\sum s_i, \mathrm{goal}$; donde goal es el objetivo de composición
t: es el tiempo que tarda el servicio s_i en emitir una respuesta al cliente
T: es el tiempo máximo que el cliente le da al servicio s_i para entregar una respuesta
d: es el dia que se llamó a ejecución al servicio s_i
h: es la hora en la que se ejecutó el servicio s_i

La reactividad de los servicios, pretende optimizar el tiempo de composición del plan utilizando aquellos servicios que tengan mejor desempeño al momento de dar una respuesta al cliente siempre y cuando sea menor a un periodo de tiempo de ejecución dado por el usuario en un lapso de tiempo discriminado por días y horas.

- Criterio 3: Manejo de Creencias

En la composición de servicios, los servicios implicados en este proceso se categorizan de dos formas. La primera son aquellos servicios que afectan el entorno en el cual se ejecuta es decir, servicios que cambian el mundo y la segunda son aquellos servicios que adquieren información del mundo sin ninguna consecuencia o afectación. Actualmente, existen compositores de servicios que utilizan las creencias para describir el mundo y componer los servicios. Estos compositores, se encargan de generar un plan de composición que satisfaga un conjunto de objetivos, para ello, busca las relaciones entre los servicios que hacen posible alcanzar dichos objetivos. Dentro de este proceso, el compositor instancia los servicios con el conjunto de posibles datos entrada y salida basado en sus creencias y envía el servicio a ser ejecutado para posteriormente comparar el conjunto de salidas esperadas por el compositor con las salidas reales del servicio después de su ejecución y, determinar si su creencia sobre dichos servicios es correcta.

El manejo de creencias en esta tesis, representará la acertada instanciación de los servicios que adquieren información del mundo durante el proceso de composición.

Definición 2.4: el manejo de creencias, determina el conjunto de posibles valores de entradas y salidas de los servicios Web más factibles asignadas por el compositor en base a sus creencias para ser utilizadas en el proceso de instanciar los servicios que intervienen en la solución de un problema de composición. Este factor, toma aquellas instancias de los SW que fueron el resultado de una ejecución exitosa del servicio.

La métrica que representa el manejo de creencias, mide la relación entre las entradas y las salidas sugeridas por el sistema de composición, y las entradas y salidas obtenidas después de la ejecución del servicio. Este proceso, permite identificar la información real que el servicio Web necesita para ejecutarse y con esto recomendar al compositor dicha información de tal manera que se mejore sus creencias, disminuyendo las malas suposiciones que este genera (ver Ecuación (2.3)).

$$MC(s_i) = \{I_C, O_C \equiv I_R\ , O_R\} \in P \mid frec(O_R) \cong 100\% \qquad (2.3)$$

Donde:
$\mathrm{I_C}, \mathrm{O_C}$: son las entradas y las salidas emitidas por el sistema de composición
$\mathrm{I_R}, \mathrm{O_R}$: son las entradas y salidas evaluadas en el mundo real después de la ejecución
P:es el plan de composición: $\sum \mathrm{s_i}, \mathrm{goal}$; donde goal es el objetivo de composición
$\mathrm{frec}(\mathrm{O_R})$:es la frecuencia que mide la relación de una entrada con una salida del servicio, aproximada al ciento por ciento de participación

El manejo de creencias está asociado directamente a la adquisición de aquella información necesaria para la ejecución del servicio Web.

Como caso de estudio, un servicio requiere de una entrada que es el código del artículo y retornar como salida el nombre y la marca del artículo. El compositor supone que al tener como entrada el código artículo 12303, las salidas serán Televisor de marca LG sin embargo al ejecutar el servicio en el mundo real, cuando con la entrada 12303, las salidas son MP3 de marca sony. Las creencias del compositor son erróneas y por lo tanto se sugiere al compositor las salidas más probables para su ejecución.

2.2 Riesgo y factores de impacto

Para determinar el riesgo de falla en la composición de servicios, hay que diferenciar entre el riesgo ocasionado por el sistema de composición y el riesgo del entorno de los servicios. El primero, analiza el conjunto de servicios instanciados y determina cuales son los apropiados para ser ejecutados según sus instancias, obteniendo el número de instanciaciones posibles para el servicio del total de instanciaciones. Mientras que el segundo, se evalúa a través de la combinación de dos criterios: disponibilidad y reactividad.

La evaluación del riesgo se realiza analizando la participación de cada uno de los servicios dentro del plan de composición con el fin de encontrar el valor del riesgo asociado al servicio.

Así, la evaluación del riesgo al ser la integración de los dos criterios, se utiliza un valor de impacto ω, el cual le asigna un valor de importancia a cada criterio (ver Ecuación (2.4)).

$$Riesgo(s_i) = \omega_1 * \big(1 - D(s_i)\big) + \omega_2 * \big(1 - R(s_i)\big) |\ \omega_1 + \omega_2 = 1, i = 1,2, \dots, n \qquad (2.4)$$

Donde:
s_i: es el servicio ejecutado
ω_1 y ω_2 : son los valores de impacto para cada criterio
$D(s_i)$: es la disponibilidad del servicio s_i
$R(s_i)$: es la reactividad del servicio s_i

En los trabajos estudiados en el estado del arte, obtener un valor conjunto de riesgo está relacionado con el análisis de decisión multicriterio (MCDA) (Triantaphyllou, 2000), el cual permite valorar diferentes características de calidad del servicio y obtener por cada característica un valor final para el plan de composición dado que la valoración requiere de 2 características (disponibilidad y reactividad), es necesario definir un impacto que determine cuál es la característica que tendrá mayor relevancia para la medición final. Esta asignación del impacto, se establece en la mayoría de los casos por el usuario final. Sin embargo, dicha asignación incurre en un alto grado de subjetividad en el análisis ya que el error estaría dado en función de experiencia y/o experticia del usuario.

Para tratar de disminuir el error de subjetividad en el análisis, en este trabajo se establece el uso de la entropía como heurística estadística de multicriterio, para determinar cuál es el criterio C_i de mayor relevancia para tener el impacto más alto y así evaluar el riesgo de falla.

Las ecuaciones 2.5 y 2.6 (Ahmed & Gokhal, 1989), determinan el valor de entropía para los criterios disponibilidad y reactividad de los servicios. La primera, establece el valor de entropía de cada criterio y la segunda permite evaluar el comportamiento de los valores asociados a cada criterio, es decir sus porcentajes de riesgos de falla.

El cálculo de estos datos se expone en el anexo A de este trabajo tesis.

$$I(C_i) = \sum_{j=1}^{nv(C_i)} \frac{n_{ij}}{n}\ l_{ij} \quad (2.5)$$

$$l_{ij} = -\sum_{k=1}^{nc} \frac{n_{ijk}}{nij}\ log_2\ \frac{n_{ijk}}{nij} \quad (2.6)$$

Donde:
n_{ijk} : es el total de registros de la clase i
n_{ij} : es el total de registros
c: son los valores posibles

De acuerdo a la experimentación presentada en el anexo A, se determinó que el criterio con mayor relevancia para valorar el riesgo de falla en composiciones de servicios es la disponibilidad por lo tanto, será este criterio el que tenga un mayor impacto en el cálculo de riesgos, que a diferencia de los trabajos revisados en la literatura, se evita la subjetividad que se presenta al dejar calificar al usuario los criterios y de esta forma comprometer los resultados finales para la selección de dichos criterios.

Ahora bien, es necesario determinar cuál será el valor del impacto para cada criterio. Para esto, la literatura presenta un conjunto de heurísticas para obtener este valor (Saeid, Abd Ghani, & Selamat, 2011). Estos son: rangos ordenados por centróide (ROC) (ver, Ecuación (2.7)), rangos recíprocos (RR) (ver, Ecuación (2.8)) y suma de rangos (RS) (ver, Ecuación (2.9)).

$$w(i) = \frac{1}{m}\sum_{k=1}^{m} \frac{1}{k}\ , i = 1,2, \dots, m \quad (2.7)$$

$$w(i) = \frac{\frac{1}{i}}{\sum_{j=1}^{n} \frac{1}{j}} =,\ i = 1,2, \dots, m \quad (2.8)$$

$$w(i) = \frac{m+1-i}{\sum_{k=1}^{m} k} = \frac{2(m+1-i)}{m(m+1)}, i = 1,2, \dots, m \quad (2.9)$$

Donde:
m: es el número de criterios

Estas heurísticas evalúan el conjunto de criterios teniendo en cuenta un valor de importancia para cada uno de ellos. Como se menciona anteriormente, este valor es asignado de forma subjetiva, mientras que en este trabajo, la valoración de la entropía permite determinar cuál es la jerarquía de importancia de los criterios.

Para la selección de la heurística que representa mejor el riesgo de falla, se realizó un experimento descrito en el anexo A, el cual dio como resultado que la heurística que se adapta mejor para calcular los impactos a los criterios es rangos recíprocos (RR).

Tal y como se demuestra en el anexo A, el impacto más alto es asignado al criterio de disponibilidad (0.67), mientras que el impacto más bajo es asignado a la reactividad de los servicios (0.33).

Finalmente, el riesgo de la composición p, es una función de optimización en la cual, se busca minimizar el valor del riesgo (ver Ecuación (2.10)).

$$Riesgo(p) = Min(\prod_{i=1}^{n} Riesgo(s_i)) \quad (2.10)$$

Donde:
$Riesgo(s_i)$: es el riesgo de cada servicio

2.3 Discusión

En este capítulo, se presentaron los criterios que esta tesis utilizará para evaluar las características particulares de los servicios, con el fin de obtener información relevante para hacer un análisis de riesgos y de esta manera, hacer una representación del manejo de fallas por parte de los servicios Web, durante su composición. Implementando el uso de periodos de tiempo para la evaluación del comportamiento los servicios y las fallas ocasionadas por el compositor.

Los criterios seleccionados en este capítulo, representan los dos tipos de riegos de falla que evalúa esta tesis: riesgo de falla del entono de los servicios y riesgo de falla del entorno del compositor. Los criterios que valoran el primer riesgo son la disponibilidad y la reactividad de los servicios enfocando su información en periodos de tiempo que se encuentran discriminados en días y horas.

El otro riesgo de falla asociado al entorno del compositor, es analizado por el manejo de creencias y su certeza con el mundo real. Este criterio, evalúa las creencias del sistema de composición a través del análisis de las entradas y salidas del servicio Web al momento de su ejecución y evalúa el estado de creencia del sistema de composición a partir de su valor de certeza asociado a las instancias del servicio, para determinar cuál es el riesgo que se presenta de acuerdo a la mala instanciación de datos. Esto con el fin de mejorar el conocimiento del compositor de servicios porque existen compositores que trabajan con esta información y la creencia de las mismas.

3. Técnicas de aprendizaje en la composición de servicios Web

Este capítulo, proporciona una visión general respecto a las técnicas de aprendizaje de máquinas utilizadas en la adquisición de información para composición de servicios Web y se muestra cuales son las técnicas que mejor representa la adquisición de la información relacionada con el factor de riesgo de falla descrita en el capítulo 2.

3.1 Introducción al aprendizaje de máquinas

El aprendizaje de máquinas es una temática que se enfoca en la adquisición de nueva información a partir de un conjunto de datos. Dependiendo del contexto que se necesite, esta temática contiene un conjunto de técnicas de aprendizaje capaces de tomar y manipular datos del entorno y trasladarlos a un conjunto de información con el fin que un sistema tome mejores decisiones y optimice su actuación en el futuro.

Aunque cada una de las técnicas de aprendizaje tiene una forma de presentar, asociar e inferir un conocimiento, éstas también tienen un conjunto de ventajas y desventajas sobre las otras que reflejan su complejidad o viabilidad, lo que hace posible identificar cual es la técnica más apropiada para ser utilizada bajo un contexto particular.

Dentro de la composición de servicios, existe un conjunto de datos que se analizan con las técnicas de aprendizaje. En primer lugar, el entorno dinámico de la Web se convierte en una fuente de datos para analizar comportamientos de los servicios. El monitoreo constante de estos datos, se estudia para identificar sus patrones de comportamiento, de manera que se adquiera la información desconocida por el compositor pero que haga posible mejorar su funcionamiento. En segundo lugar, el monitoreo de los servicios también proyecta información del sistema de composición la cual contribuye al mejoramiento en la toma de decisiones al interior de los procesos de composición.

3.2 Análisis de las técnicas de aprendizaje

Cada una de las técnicas descritas en la sección 1.4, presenta una serie de ventajas y desventajas que las identifica y las diferencia de las otras. Asimismo, comparten algunas características relevantes para el proceso de aprendizaje de riesgos de falla.

A continuación, se presenta una tabla comparativa en la que se describen las ventajas y desventajas de cada técnica de aprendizaje (**Tabla 3-1**).

Tabla 3-1: Técnicas de Aprendizaje – Características -

Técnicas de aprendizaje	Ventajas	Desventajas
Arboles de Decisión (Qulnlan, 1986), (Quinlan, 1993), (Araujo Sierra, 2006), (Russell & Norving, 2004), (Mitchell, 1997)	• El lenguaje de representación es claro. • Maneja interacciones complejas entre variables. • Utiliza un modelo de caja blanca. Si un determinado resultado es proporcionado por un modelo, la explicación para el resultado es fácilmente comprensible. • Se combina con otras técnicas de decisión • Son relativamente fáciles de entender cuando hay pocas decisiones • Son muy robustos cuando se trabaja con valores extremos.	• Llega a generar árboles largos y complejos que son más difíciles de comprender y explicar. • La complejidad del problema crece casi linealmente y su entrenamiento es muy costoso computacionalmente.
Programación lógica Inductiva (ILP) (Muggleton, 1991), (Muggleton & Raedt, 1994) (Lavrac & Saso, 1994), (Kersting, 2006), (Mitchell, 1997), (Araujo Sierra, 2006)	• El lenguaje expresivo para codificar los dos modelos y datos. • La información proporcionada proviene de diferentes fuentes. • Combina cálculos numéricos con datos relacionales. • Comúnmente utilizados en la Web ya que poseen métodos para representar el contenido Web y las relaciones entre palabras a través de representaciones relacionales. • Permite hacer uso de un conocimiento previo para mejorar la selección de la hipótesis.	• Lento para aplicaciones complejas • El espacio de búsqueda crece "muy rápido" con la cantidad de información proporcionada en el conocimiento previo. • Si la selección del conocimiento previo es equivocada ocasiona la generación ineficiente de la hipótesis y prolongar el proceso de selección de la misma.
Razonamiento basado en Casos (CBR) (Aamodt & Plaza, 1994), (Kolodner, 1983), (Kolodner, 1993)	• Proponen soluciones rápidamente. • No necesita conocer completamente el dominio. • Casos son útiles para conceptos mal definidos • La facilidad de explicación: los resultados de un sistema CBR se justifican basándose en la similitud de los problemas actuales en el caso recuperado. • Actúa con un conjunto mínimo de casos resueltos adquiridos en una base de casos y los nuevos casos son almacenados en la base de casos para futuros procesos.	• Los casos viejos llegan a ser obsoletos. • Los casos más apropiados no son recuperados • Necesita conocimiento para realizar la adaptación • Tiene problemas con dominios dinámicos donde son incapaces de seguir un cambio en la manera en que se resuelven los problemas, ya que suelen ser fuertemente sesgado hacia lo que ya ha trabajado. Esto resulta en una base de casos obsoletas. • No tolerancia a datos que

Técnicas de aprendizaje	Ventajas	Desventajas
		ocasionan ruido lo que implica almacenamiento de casos innecesarios. Esto implica el almacenamiento y recuperación ineficiente de los casos. • En ocasiones los sistemas CBR requieren información aportada por los usuarios.
Aprendizaje por refuerzo (Kaelbling, Littman, & Moore, 1996), (Araujo Sierra, 2006), (Mitchell, 1997)	• Soluciona cada subproblema exactamente una sola vez. • Guarda soluciones parciales dentro de una tabla. • Tiene un menor costo de ejecución que los algoritmos recursivos. • Tomar ventaja del traslape de subproblemas. • Requiere menos conocimiento a priori, y se aplica donde el aprendizaje supervisado no es aplicable	• En muchas ocasiones la calidad de la solución obtenida no es satisfactoria y es muy lenta en converger • Elevado costo en memoria para almacenar los valores de cada estado, ya que los problemas son muy complejos. • En ocasiones se aprende todo de nuevo • Por la percepción limitada, a menudo es imposible determinar con toda certeza el estado actual. Esto afecta el desempeño del algoritmo
Aprendizaje Bayesiano (Sebe, Cohen, Garg, & H., 2005), (Russell & Norving, 2004), (Araujo Sierra, 2006), (Mitchell, 1997)	• Se basa en las estimaciones máximas verosímil o máximo a posteriori. • Utiliza el algoritmo de búsqueda estructural. • Permiten aprender sobre relaciones de dependencia y causalidad. • Permiten combinar conocimiento con datos. • Evitan el sobre-ajuste de los datos. • Maneja bases de datos incompletos	• Utiliza un algoritmo amplio de búsqueda para aprender el modelo y para clasificar nuevos casos llega a ser más complejo y pierde la simplicidad. • Trabajar con datos a priori y posteriori Sin embargo la asignación de especificaciones a priori son difíciles de generar. • La asignación del número real para cada ajuste de los parámetros del modelo del mundo se vuelve complicado si no se tiene un amplio conocimiento del lenguaje para describir las especificaciones a priori o en ocasiones estos datos llegan a ser inventados es decir colocar datos por conveniencia. • Requiere un coste

Técnicas de aprendizaje	Ventajas	Desventajas
		computacional alto independiente si se trabaja con especificaciones a priori o posteriori

Si bien es cierto todas las técnicas de aprendizaje descritas en la sección 1.4, se utilizan para adquirir diferente información asociada al riesgo de falla, en el marco de esta tesis, se han seleccionado aquellas que cumplan con unas características en particular:

- Manejo de recursos de memoria, almacenamiento y tiempo de ejecución
- Información requerida
- Representación
- El resultado se interprete fácilmente por el humano

Teniendo en cuenta el análisis realizado en la sección 2.1 de esta tesis, las variables de riesgo a calcular son: manejo de creencias, la disponibilidad de servicios en periodos de tiempo y restricciones de tiempo en ejecuciones de servicios en periodos de tiempo.

De acuerdo a la primera característica, las cinco técnicas de aprendizaje dependen del número de ejemplos que utilicen ya que estos incrementan el espacio de hipótesis y por lo tanto, los requerimientos de hardware se verán afectados. Sin embargo, en algunos casos como el aprendizaje por refuerzo y el CBR necesitan más información para evaluar un estado y por lo tanto el incremento de requerimientos se hace necesario es decir, más manejo de memoria y de almacenamiento a diferencia de las otras técnicas de aprendizaje.

La segunda característica determina que información es la necesaria para adquirir nueva información. En la técnica de CBR, este proceso se complica cuando no se determina cual es la información necesaria que lleva cada caso. Por lo tanto, este proceso genera resultados inexactos e incluso casos con información inútil para el proceso. Por otro lado, el aprendizaje bayesiano requiere en ocasiones probabilidades a priori que son complicadas de calcular al inicio del proceso.

La teoría (Araujo Sierra, 2006), (Dietterich, 2003), (Mitchell, 1997), (Blockeel & Raedt, 1998) o, (Russell & Norving, 2004) indica que los árboles de decisión se caracterizan por su facilidad de representación del su lenguaje; además, su interpretación y manipulación permiten agilizar el diseño y modelado del sistema.

En esta tesis, se considerará que el aprendizaje no solo es para las máquinas sino también que el humano comprenda el porqué o de donde viene la información de aprendizaje y que él, también tenga conocimiento de ella. Teniendo en cuenta lo anterior, el aprendizaje basado en árboles de decisión y la programación lógica inductiva alcanzan esta característica fácilmente. El primero, actúa como una caja blanca donde es posible conocer todo el proceso del tratamiento que se les da a los datos. Y el segundo, la ILP proporciona inferencias lógicas como resultado las cuales son fácilmente comprendidas por los humanos.

En este punto es importante aclarar que algunas de las técnicas presentadas en la anterior sección, interactúan entre sí para mejorar los cálculos en la selección de una hipótesis y mejorar los resultados. Es así como la integración se da en el aprendizaje bayesiano, aprendizaje por refuerzo o ILP utilizando la técnica de árboles de decisión.

Por las características y la descripción realizadas en el marco de esta tesis, la técnica de aprendizaje de máquina que será utilizada es: los árboles de decisión lógicos, con el fin de obtener automáticamente la información relacionada a las fallas en composiciones de servicios a través de una forma lógica y comprensible para el humano.

3.3 Discusión

En este capítulo, se presentaron las diferentes técnicas de aprendizaje de máquinas que según el estado del arte son las más utilizadas en los procesos de composición de servicios.

El análisis previo, permitió identificar las ventajas y desventajas que presenta cada una de estas técnicas con el fin seleccionar aquellas técnicas que mejor representen el modelo de adquisición de la información referente al manejo de fallas teniendo en cuenta el sistema de composición utilizado. Para ello, se establecieron las características de evaluación para cada técnica y así seleccionar la mejor.

- Manejo de recurso memoria, almacenamiento y tiempo de ejecución
- Información requerida
- Representación
- El resultado se interprete fácilmente por el humano

El análisis hizo posible identificar que las técnicas que mejor describieron las características anteriormente mencionadas son los árboles de decisión y la programación lógica inductiva. Sin embargo, estas dos técnicas se encuentran fusionadas dentro del marco de los árboles de decisión lógicos. Por lo tanto, esta técnica será adaptada en el modelo de aprendizaje propuesto en el capítulo 5 para adquirir la información referente a los riesgos de falla de los servicios durante las composiciones Web.

4. Modelo de aprendizaje de máquinas para el manejo de riesgos de falla

En este capítulo, se presenta el modelo de aprendizaje de máquinas para adquirir de manera automática la información relacionada con el riesgo de falla en las composiciones de servicios y adicionalmente, se detalla el mecanismo de integración del modelo de aprendizaje con técnicas de planificación.

4.1 Visión general del modelo de aprendizaje

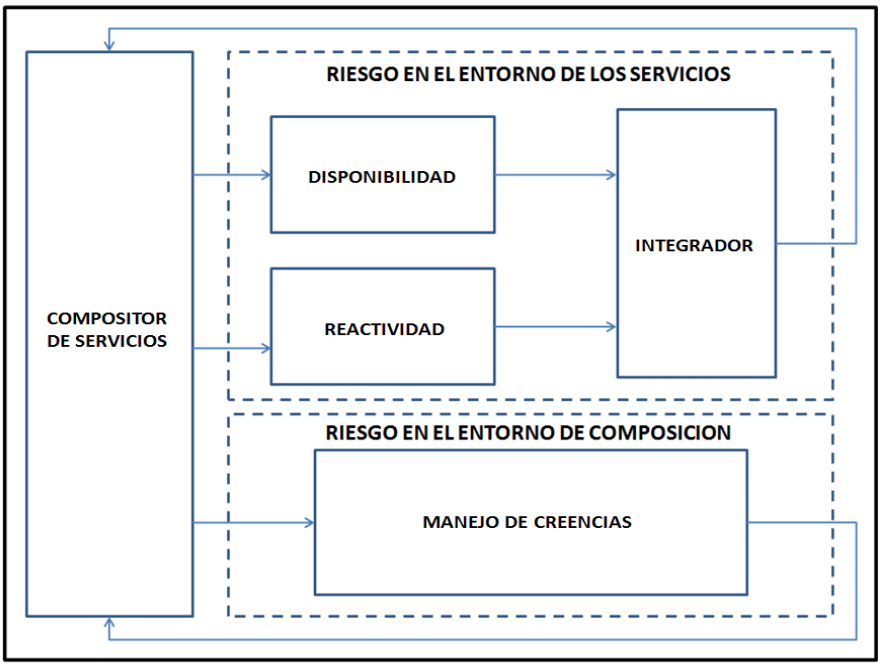
El problema de saber qué composición de servicios es la mejor desde el punto de vista de su falla en esta tesis es analizado a través de la disponibilidad y la reactividad de los servicios junto con el manejo de las creencias del compositor.

En primer lugar, la disponibilidad y la reactividad determinan el riesgo de falla dentro del entorno de los servicios. La valoración consiste en un análisis del comportamiento real del servicio ejecutado como el generador de los riesgos es decir, que de su comportamiento depende el éxito de un plan de composición por lo tanto, se hace necesario que el compositor conozca esta información. En segundo lugar, el manejo de creencias del compositor evalúa el entorno del compositor. Esto, hace referencia a la evaluación de las suposiciones en cuanto al manejo de las entradas y salidas de cada servicio Web que el sistema de composición asume como buenas dentro de sus creencias pero que al ser evaluadas en el contexto real, éstas coinciden o no con las supuestas por el compositor.

A continuación, se presenta el modelo de aprendizaje que se propone para manejar el riesgo de fallas con las características anteriormente descritas el cual se muestra en la figura 4-1. Este modelo, está compuesto de cuatro módulos básicos:

- Módulo de Disponibilidad
- Módulo de Reactividad
- Módulo de Manejo de Creencias
- Módulo de Integración

Figura 4-1: Modelo de aprendizaje de riesgos de falla propuesto.



Los módulos de disponibilidad, reactividad y manejo de creencias, hacen uso de dos etapas, las cuales les permiten obtener el factor de riesgo para cada servicio. Estas etapas son:

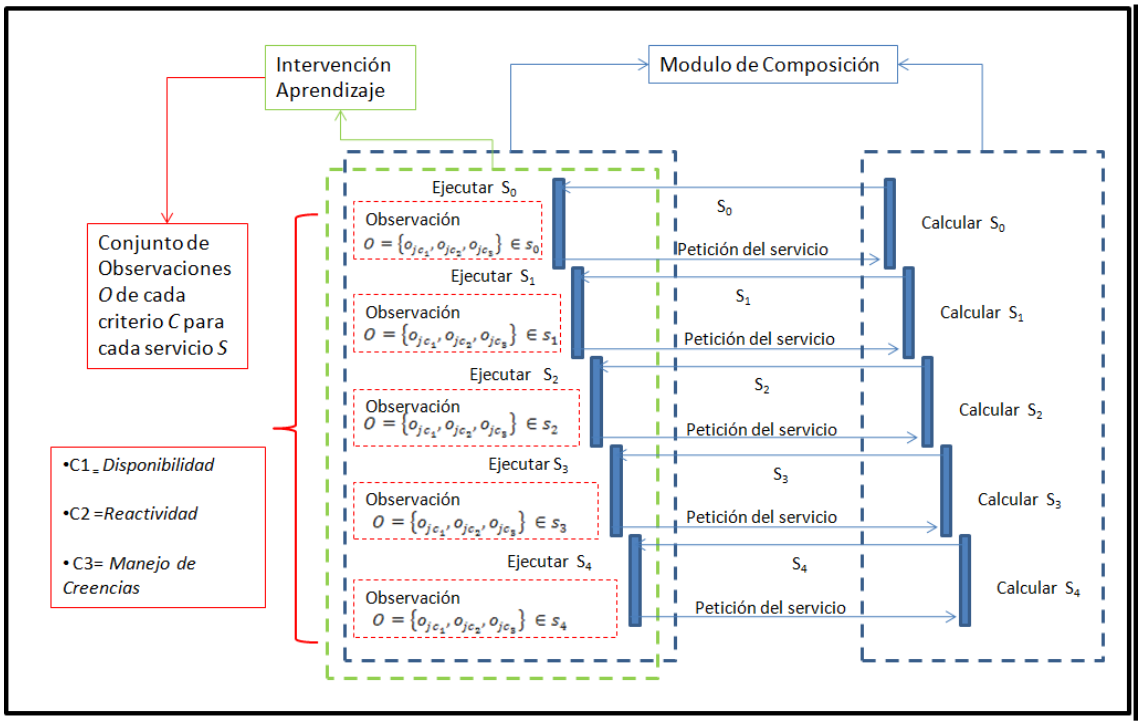
- Etapa 1: Adquisición y Traducción de la Información de Ejecución

Se refiere a la captura de la información de los servicios a partir de observaciones generadas en ejecución de dichos servicios como se muestra en la figura 4-2. En esta figura, se observa que el módulo de composición calcula de manera incremental aquellos servicios que permite alcanzar un subojetivo de composición y como éstos, son enviados uno a uno a ejecución para saber si este servicio hace parte del plan final de composición o no. Cuando un servicio s_i es ejecutado, se está evaluando por un lado las creencias del compositor respecto al mundo y por otro lado, se captura la información de cada servicio tras su ejecución mediante el uso de un monitor perteneciente al módulo de aprendizaje el cual, por cada criterio $c_k \in C$ seleccionado, se genera un conjunto de observaciones $O_j = \{o_1, o_2, o_3 \dots o_n\}$, donde cada una de las observaciones del conjunto, almacena el servicio $s \in S$ que se ejecutó y, el conjunto de datos característicos del criterio para ese servicio.

Por lo tanto, cada vez que el servicio es ejecutado, la lista de observaciones O_j se incrementará $O_j = \{o_1, o_2, o_3, \dots \dots, o_n\}$ y de esta manera se obtiene la información sobre el comportamiento del servicio evaluado bajo un determinado entorno de ejecución.

$$\forall\, s_i \in S \mid \exists\, c_k \rightarrow O_j\{o_1, o_2, o_3, \dots \dots, o_n\}\; i = 1{,}2 \dots . n, k = 1{,}2{,}3$$

Figura 4-2: Modelo de ejecución.



Cuando este proceso termina, el módulo de composición selecciona un nuevo servicio para continuar alcanzando los objetivos y construir un plan de composición como un proceso incremental hasta alcanzar todos los objetivos.
Posteriormente, por cada criterio $c_k \in C$ evaluado, un traductor es el encargado de interpretar cada una de las observaciones y traducirlas a un lenguaje L de representación del aprendizaje en el cual, se identifica el conjunto de ejemplos de entrenamiento haciendo referencia a la base de conocimientos kb y a la función objetivo τ, es decir, qué es lo que se quiere aprender a partir de la base de conocimientos kb.

- Etapa 2: Generación del Conocimiento de Control

Esta etapa, consiste en que por cada servicio $s_i \in S$ el componente de aprendizaje construye un árbol de decisión lógico T, donde cada nodo contiene las condiciones bajo las cuales es aplicado cada criterio c_k y los nodos hoja del T, contienen los valores de predicción para cada criterio c_k. La estimación de la probabilidad de las reglas es obtenida de la frecuencia de los ejemplos de aprendizaje.

La construcción del árbol se realiza con el sistema TILDE el cual implementa el algoritmo para la inducción de árboles de decisión lógicos presentado en la figura 4-3 (Blockeel & Raedt, 1998).

Figura 4-3: Algoritmo para inducir árboles de decisión lógicos.

```
1.  procedure builtree(T, ε, Q)
2.      If ε homogeneus enough then
3.          K:= most_frecuent_class ε
4.          T:= leaf (k)
5.      Else
6.          Q_b ≔ best element of p(Q), according to some heuristic
7.          T.test: C' where Q_b =← Q, C
8.          ε_1 ≔ {E ∈ ε |E ∪ β| = Q_b}
9.          ε_2 ≔ {E ∈ ε |E ∪ β| ≠ Q_b}
10.         buildtree(T.left, ε_1, Q')
11.         buildtree(T.right, ε_2, Q)
```

En este algoritmo, la línea 2 verifica si el conjunto de ejemplos de entrenamiento ε son homogéneos, si esto es cierto, se obtiene la clase K más frecuente de ε (línea 3) y se genera la rama izquierda del árbol asociándole la clase y el valor de la frecuencia de los ejemplos (línea 4). Si el conjunto de ejemplos no son homogéneos, se asigna el mejor elemento de clasificación (línea 6). En la línea 7, se asignan el conjunto de ejemplos en los cuales se encuentra el elemento seleccionado mientras que en la línea 8, son asignados los ejemplos que no cumplen esta condición. Finalmente, las líneas 9 y 10, crean las ramas izquierda y derecha del árbol a partir de la asignación del elemento seleccionado para la clasificación.

Como resultado de aplicar las técnicas de aprendizaje de máquinas, se obtiene un conjunto de reglas de aprendizaje $R\{r_1, r_2, r_3, \ldots\ldots, r_n,\}$ que representan el comportamiento de cada servicio s_i teniendo en cuenta el criterio de evaluación c_k.

Finalmente, el conjunto de reglas generadas por el sistema de aprendizaje constituyen el conocimiento de control, el cual es trasladado a una base de datos (knowledgeDB) para ser utilizado en futuras composiciones.

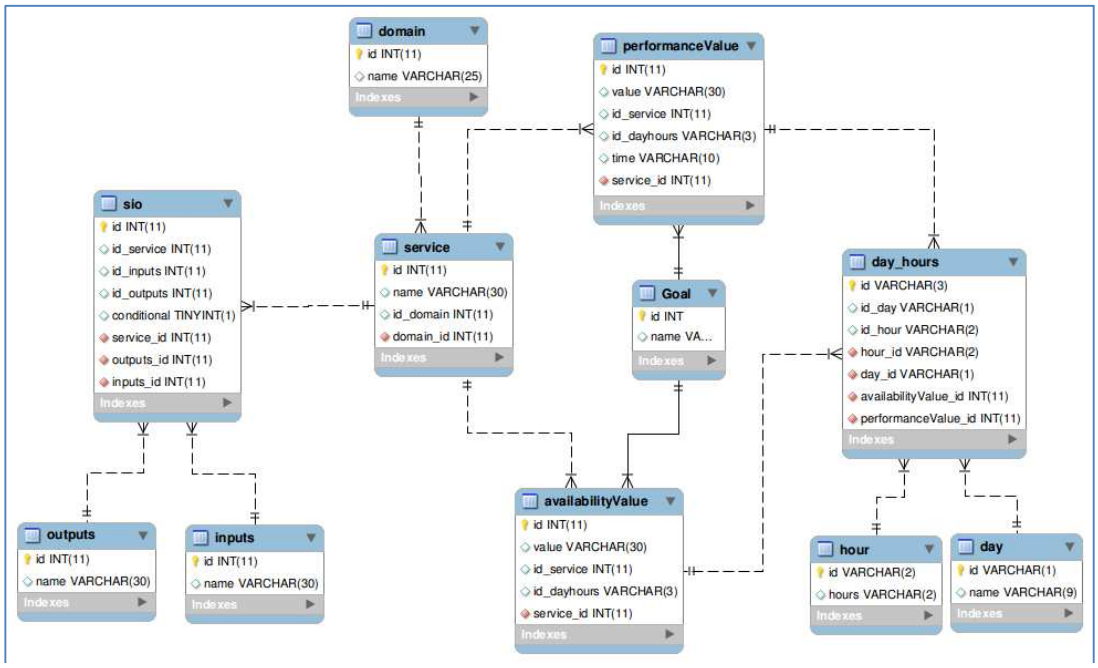
El diagrama relacional de la base de datos knowledgeDB se presenta en la figura 4-4.

Esta base de datos, hace uso de las tablas availabilityValue, performanceValue y sio como medio de almacenamiento del nuevo conocimiento asociado a los criterios de disponibilidad, reactividad y manejo de creencias de los servicios adquiridos a través de las reglas de aprendizaje.

Sin embargo, para hacer uso de estas tablas es necesario contar con una información adicional que involucra el uso de las otras tablas de la base de datos. Para las tablas availabilityValue y performanceValue, es necesario referenciar el servicio Web al que está asociada la regla de aprendizaje para esto, se hace uso de la tabla service. Asimismo, es necesario identificar la hora y el día descritos en la regla de aprendizaje y referenciarla en las tablas anteriormente mencionadas, para ello se utiliza las tablas day y hour relacionadas en la tabla day_hours. Finalmente el objetivo de composición perteneciente a la regla de aprendizaje se almacena en la tabla goal y al igual que las anteriores, ésta también es referenciada en las tablas availabilityValue y performanceValue. Por otro lado, la tabla sio a parte de la referencia de la tabla servicio requiere de la información de las entradas y salidas del servicio, las cuales son

almacenadas en las tablas inputs y outputs para finalmente ser referenciadas en dicha tabla.

Figura 4-4: Diagrama relacional base de datos KnowledgeDB



El último módulo, es el de integración, aquí, el conocimiento de control es interpretado para hacer parte de la información requerida por el sistema de composición de manera que exista la unión entre los procesos. El algoritmo de integración, se presenta en la figura 4-5:

Figura 4-5: Algoritmo de integración de datos.

1. $\boldsymbol{While}\ \exists\ s_i \in S\ \boldsymbol{do}$
2. $profile \leftarrow readProfile(s_i)$
3. $assignQoS(c_k, profile)$
4. $write(newOntology) | newOntology \leftarrow newprofile(s_i) \cup ontology(s_i)$
5. $\boldsymbol{end\ while}$

6. $\boldsymbol{procedure}\ assignQoS(c_k, profile)$
7. $avil \in qosontology\ class$
8. $avid \in qosontology\ Individual$
9. $//crea\ una\ instancia\ de\ la\ clase\ avil$
10. $avid \leftarrow Inividual(c, avil)$
11. $//crea\ las\ propiedades\ de\ impacto\ y\ riesgo\ para\ el\ criterio\ con\ sus\ respectivos$
12. $//valores$
13. $avid \leftarrow Property(\text{non\_funtional\_parameterHasValue}, \text{value})$
14. $avid \leftarrow Property(\text{non\_funtional\_parameterHasWeight}, \text{value})$
15. $//\ asocia\ al\ perfil\ del\ servicio\ un\ parámetro\ y\ las\ instancias\ de\ la\ ontología\ qos$
16. $profile \leftarrow serviceParameter \wedge avid$
17. $newProfile(s_i) = profile$

En este algoritmo, la nueva ontología es una modificación del perfil de cada servicio s_i de la siguiente manera: la línea 2, captura el perfil del servicio profile y a través de la función assignQoS (líneas 3-6), se asigna una instancia avid a la ontología QoS (línea 8). Esta instancia, representa el criterio c_k del riesgo. Adicionalmente a esa instancia, se le crean dos propiedades que representa el valor del riesgo para el criterio c_k y su valor de impacto asociado (líneas 13-14). Finalmente, se asocia al perfil del servicio un parámetro serviceparameter con las instancias de la ontología qos avid (línea 16) obteniendo el nuevo perfil del servicio newProfile(s_i) (línea 17). Finalmente, la línea 4 crea el archivo de la nueva ontología del servicio s_i.

A continuación se describe cada uno de los módulos del modelo de aprendizaje.

4.2 Módulo de disponibilidad

Este módulo es el encargado de adquirir la información de riesgo según el criterio de disponibilidad, requiere como entradas las observaciones de la ejecución con respecto a la disponibilidad de los servicios en periodos de tiempo representado en días y horas semanales y produce como salida un conjunto de reglas de aprendizaje por cada uno de los servicios ejecutados asociada a dicha información.

4.2.1 Adquisición de información relacionada con la disponibilidad del servicio Web

Al ejecutar un servicio Web s_i, se almacena un conjunto de datos que representan la observación O_j de monitorización del servicio, la cual contiene la tupla $(s_i, d, h, goal, cl)$ donde:
s_i: representa el servicio que se ejecutó.
d: es el día que el servicio s_i se ejecutó.
h: representa la hora en que el servicio s_i se ejecutó. Esto, ayuda a definir que tan bueno es el comportamiento del servicio en un determinado instante del día.
goal: es el objetivo a alcanzar por el plan de composición.
cl: determina si el servicio está listo para utilizarse, esto es: si el servicio no se encuentra al momento de llamarse a ejecución, se considera una falla, de lo contrario, el servicio tendrá una clasificación de éxito.

Un ejemplo de las observaciones de la disponibilidad se muestra en la figura 4-6

Figura 4-6: Observación de disponibilidad del servicio buyItem

```
Observaciones de la ejecución del servicio buyItem
Criterio Disponibilidad

buyItem,tuesday,11:00,client_posses(item12303),success.
buyItem,tuesday,17:00,client_posses(item12303),failure.
buyItem,tuesday,17:00,client_posses(item12303),failure.
buyItem,thursday,11:00,client_posses(item12303),success.
buyItem,thursday,11:00,client_posses(item12303),success.
```

En esta figura, cada fila representa una observación adquirida después de ejecutar un servicio Web. Por cada fila, se identifican 5 características particulares de cada servicio: (1) el nombre del servicio ejecutado: buyItem, (2) el día y (3) la hora en que el servicio se ejecutó: tuesday, thursday y 11:00, 17:00 respectivamente, (4) el objetivo de composición: client_posses(item12303) y finalmente (5) la clase de ejecución si fue éxito o falla: success o failure.

Este conjunto de observaciones, representa las entradas de datos para este módulo.

Registrada esta información de monitorización del servicio, ésta, se traduce al lenguaje de aprendizaje de máquinas (lenguaje lógico) mediante un algoritmo que captura la información del conjunto de observaciones O_j de cada servicio s_i . Este algoritmo y un ejemplo del lenguaje lógico donde se identifica la función objetivo (A) y base de conocimientos (B) para aprender los patrones de comportamiento de la ejecución del servicio buyItem se detalla en la figura 4-7.

Figura 4.7: Algoritmo y Ejemplo del Lenguaje Lógico del Criterio Disponibilidad del Servicio BuyItem

Algoritmo	Lenguaje Lógico
1. ***while*** $\exists\ O_j\ s_i$ ***do*** 2. $\tau \leftarrow s_i \cup cl$ 3. $kb \leftarrow d \cup h \cup goal \mid d \in day, h \in hour$ 4. ***end while***	A % Función objetivo execution_buyItem(o,class)). classes([success,failure]). B % Ejemplo o1 %Instanciación función objetivo execution_buyItem(o1,success). %Base de conocimiento tuesday(o1). 11:00(o1). client_possess{item12303}(o1). % Ejemplo o2 %Instanciación función objetivo execution_buyItem(o2,failure). %Base de conocimiento thursday(o2). 11:00(o2). client_possess{item12303}(o2).

El algoritmo descrito en la figura 4-7, permite almacenar los datos de las observaciones para cada servicio ejecutado y traslada esta información a una representación del lenguaje lógico. Para esto, se construye la función objetivo τ (línea 2) la cual, representa la variación de éxitos en el conjunto de observaciones O_j y en segundo lugar se construye la base de conocimientos kb (línea 3) que representa el conjunto de datos que permiten describir las condiciones bajo las cuales se presenta la variación de la función objetivo, esta función, está conformada por los días d y horas h en los cuales el servicio se ejecutó y el objetivo de composición $goal$. En esta figura, se observa un ejemplo de la representación del lenguaje lógico. En la parte A, se describe la función objetivo, ésta se representa por el nombre del servicio ejecutado antepuesta la palabra execution_, y asocia dos valores, el primero es un identificador demarcado con la letra o seguido, de un consecutivo esto le permite al sistema de aprendizaje enlazar la función objetivo con cada uno de los datos de la base de conocimientos para cada O_j. El segundo valor es la clase de ejecución del servicio es decir success o failure. En la parte B, se muestra la

instanciación de la función objetivo de acuerdo al dato de la clase de cada O_j. y se asocia cada uno de los datos restantes de la kb, a los cuales se les adiciona el identificador o según su observación.

Este lenguaje lógico generado, es una representación del sistema de aprendizaje Tilde el cual utiliza esta representación como datos de entrada para generar nuevo conocimiento, el que se detalla en la siguiente sección.

4.2.2 Generación del conocimiento de control para el criterio de disponibilidad

Para este criterio, se utiliza el algoritmo de árboles de decisión lógicos donde por cada servicio s_i, el algoritmo aprende un árbol Tque modela el comportamiento de servicio en términos de dos clases éxito o falla. Por cada rama del árbol de decisión, se representará una regla r_j de comportamiento del servicio correspondiente.

La figura 4-8, muestra el árbol de decisión lógico aprendido para la acción buyItem y la regla generada.

Figura 4-8. Árbol y regla de aprendizaje para el criterio de disponibilidad del servicio buyItem

```
A

        Árbol

   1    buyItem(-A,-B)
   2    hour11(A) ?
   3          +-yes:clientPossesitem12303(A) ?
   4          |        +-yes: thursday(A) ?
   5          |        |       +-yes: [success] 8.0 [[success:8.0,failure:0.0]]
   6          |        |       +-no:  tuesday(A) ?
   7                            |       +-yes: [success] 11.0 [[success:7.0,failure:4.0]]
   8                            |       +-no:  [failure] 74.0 [[success:34.0,failure:40.0]]

B

        Regla

        buyItem(A,[success:0.636363636363636,failure:0.363636363636364]) :- hour11(A),clientPossesitem12303(A),tuesday(A), !.
        % 7.0/11.0=0.636363636363636
```

En la parte A de la figura 4-8, se muestra el árbol de aprendizaje generado a partir del lenguaje lógico descrito en la sección inmediatamente anterior, aquí se observa que los nodos están representadas por las horas y los días de ejecución: 11, tuesday, thursday respectivamente y el objetivo de composición: cliente_posses(ítem_12303). Los valores de yes y no, representan la dirección de la rama del árbol es decir izquierda o derecha respectivamente las cuales ilustran la expansión del árbol de aprendizaje. Cuando el árbol llega a un nodo hoja en este caso las líneas 5,7,8 , se asignan las clases success y failure y su valor de predicción.
En la parte B de la figura 4-8, se presenta la regla de aprendizaje generada por el algoritmo Tilde a partir del árbol de aprendizaje. Ésta, es una representación Prolog del árbol en la cual, se identifica los valores porcentuales de las clases éxito y falla en la cabeza de la regla y los nodos que conducen a los nodos hoja, son el cuerpo de la regla.

Cada una de las reglas r_j calculadas en el árbol de aprendizaje, es almacenada en la base de datos, con el objetivo de mantener la información aprendida y que ésta se utilice para integrarse con el modelo de composición en futuras composiciones. Estas reglas, están compuestas por el día y la hora de ejecución y el objetivo de composición. Cada regla contiene un valor que representa el valor individual del riesgo del servicio. El algoritmo utilizado para traducir y almacenar las reglas se describe en la figura 4-9.

Figura 4-9 Algoritmo de traducción y almacenamiento de reglas de aprendizaje para el criterio de disponibilidad

1. **while** $\exists\, r_j \in R(s_i)$ **do**
2. $\forall\, r_j \exists\, d \cup h \cup goal \mid d \in day, h \in hour$
3. $r.value = number$
4. **if** (r_j)**then**
5. $knowledgeDB \leftarrow r_j$
6. $availabilityValue(s_i) \leftarrow r_j.value$
7. **endif**
8. **endwhile**

En la línea 2 del algoritmo, se especifica que las reglas generadas por el aprendizaje de máquinas, tiene un día, una hora y un objetivo de composición. La línea 3 adquiere el valor porcentual de la regla. Si existe una regla con dichas características (línea 4), la regla se crea en la base de datos y el valor de la regla que representa su riesgo se almacena en la tabla availabilityValue de la base de datos (línea 6).

4.3 Módulo de reactividad

En este módulo, la información requerida como entrada, es el conjunto de observaciones que representa la reactividad de los servicios con información temporal, representada mediante el día y la hora de dicha observación. A partir de ésta, se obtiene como salida las reglas lógicas de aprendizaje $r_j \in R$ de los servicios en función de los días y las horas teniendo como base, el tiempo que este requiere para emitir una respuesta por parte del servicio hacia el cliente y el objetivo de composición.

4.3.1 Adquisición de información relacionada con la reactividad del servicio Web

Cuando el servicio s_i se ejecuta, es posible que el usuario imponga una restricción del tiempo te que debe tardar el servicio en responder a una petición es decir, desde el instante que el cliente envía la solicitud, hasta que recibe la respuesta a dicha solicitud. Con esto, es posible determinar la eficiencia del servicio para responder a restricciones de tiempo en su ejecución. Asociado a esto, es posible identificar la hora y el día en que

se ejecutó para obtener un mejor nivel de detalle cuando actúa el servicio. La observación que se almacena para este caso viene dada por la tupla $(s_i, te, d, h, goal, cl)$.

Donde:
s_i: es el servicio que se ejecutó.
te: es la restricción de tiempo para la ejecución de un servicio asignada por el cliente.
d: es el día que el servicio s_i se ejecutó.
h: representa la hora en que el servicio s_i se ejecutó.
$goal$: es el objetivo a alcanzar por el plan de composición.
cl: es la clasificación que tiene un servicio al ser ejecutado. Esto, está relacionado con el tiempo que tarda un servicio al ejecutarse. La clasificación es exitosa si el servicio alcanza a ejecutarse en el periodo de tiempo restrictivo emitido por el cliente, de lo contrario su ejecución será fallida.

En la figura 4-10 se presenta un ejemplo de las observaciones referentes a la reactividad de los servicios.

Figura 4-10: Observación de reactividad del servicio BuyItem

```
Observaciones de la ejecución del servicio buyItem
Criterio Reactividad

buyItem,300,tuesday,11:00,client_possess(item12303),failure.
buyItem,200,thursday,17:00,client_possess(item12303),success.
buyItem,600,tuesday,17:00,client_possess(item12303),failure.
buyItem,300,thursday,11:00,client_possess(item12303),success.
buyItem,300,thursday,11:00,client_possess(item12303),success.
```

Esta figura, contiene cinco observaciones del servicio buyItem separadas por un salto de línea y donde cada una contiene 6 datos particulares separados por ‘,’. El primer dato encontrado es el (1) nombre del servicio: buyItem, seguido de la (2)restricción de tiempo de ejecución impuesta por el usuario y medida en milisegundos: 300, 200, 600, además, el (3)día y la (4)hora que el servicio se ejecutó: tuesday, Friday y sunday y 11:00, 9:00, 17:00, 20:00, 21:00 respectivamente, (5)el objetivo de composición: client_posses(item12303) y finalmente la (6)clase de ejecución del servicio es decir, éxito o falla: success o failure. Cada una de estas observaciones, hacen parte de las entradas de este módulo.

A partir del registro de estas observaciones, éstas, se trasladan a un lenguaje lógico a través de un algoritmo que toma las observaciones O_j de cada servicio s_i. Este algoritmo y un ejemplo del lenguaje lógico se presentan en la figura 4-11

Figura 4-11: Algoritmo y ejemplo del lenguaje lógico del criterio reactividad del servicio buyItem

Algoritmo	Lenguaje Lógico
1. ***while*** $\exists\ O_j\ s_i$ ***do*** 2. $\tau \leftarrow s_i \cup cl$ 3. $kb \leftarrow d \cup h \cup te \cup goal \mid d \in day, h \in hour$ 4. ***end while***	A % Funcion Objetivo execution_buyItem(o,class). classes([success,failure]). B % Ejemplo o1 %Instanciación función objetivo execution_buyItem(o1,300). %Base de conocimiento 300(o1). tuesday(o1). 11:00(o1). client_possess{item12303}(o1). % Ejemplo o2 %Instanciación función objetivo execution_buyItem(o2,200). %Base de conocimiento 200(o2). thursday(o2). 17:00(o2). client_possess{item12303}(o2).

El algoritmo presentado en la figura anterior, permite almacenar las observaciones relacionadas con la reactividad de los servicios y las traslada a una representación de un lenguaje lógico utilizado por el sistema Tilde para la generación del nuevo conocimiento. En primer lugar, este algoritmo genera la función objetivo τ a través de la line 2, esta recibe los datos del nombre del servicio y la clase de ejecución del servicio es decir, éxito o falla. La línea 3, genera la base de conocimientos kb, ésta contiene los datos bajo los cuales se aplica la función objetivo. Los datos asignados a la base de conocimientos son: los días d y horas h de ejecución del servicio, la restricción de tiempo de ejecución del servicio te y el objetivo de la composición goal.

La figura 4-11, también muestra un ejemplo del lenguaje lógico. En la parte (A), se describe la función objetivo la cual es construida utilizando la palabra execution_ seguida del nombre del servicio ejecutado. Ésta, asocia dos valores, el primero es un identificador demarcado con la letra o seguido de un consecutivo esto le permite al sistema de aprendizaje enlazar la función objetivo con cada uno de los datos de la base de conocimientos para cada $\mathrm{O_j}$. El segundo valor es la clase de ejecución del servicio es decir success o failure. La parte (B), muestra la instanciación de la función objetivo con las clases success y failure según sea la ejecución del servicio y se construye la representación de la base de conocimientos utilizando los datos del día y la hora de ejecución del servicio, la restricción de tiempo de ejecución y el objetivo de composición. A estos datos, se asigna el identificador o seguido del número consecutivo perteneciente a la observación.

4.3.2 Generación del Conocimiento de Control de Reactividad de los Servicios Web

Este criterio, es aprendido a través del algoritmo de árboles de decisión lógicos, donde para cada servicio, se genera un árbol que modela el comportamiento del servicio teniendo en cuenta su éxito, en función de un tiempo de restricción de ejecución de un servicio te. Cada una de las ramas del árbol generado, produce una regla $\mathrm{r_j} \in \mathrm{R}$ de aprendizaje del comportamiento del servicio $\mathrm{s_i}$

La figura 4-12, presenta el árbol de decisión lógico y la regla de aprendizaje generados por TILDE.

En la parte A de la figura 4-12 se presenta el árbol de aprendizaje generado por la herramienta de aprendizaje Tilde. En éste, se identifica los nodos con las etiquetas de la hora y los días de ejecución, el objetivo de composición y la restricción de tiempo (líneas 2,3,4,5). Por otro lado, se observa la utilización de etiquetas yes y no las cuales, permiten identificar el crecimiento del árbol. Los valores asociados a la clases éxito y falla son los nodos hoja los cuales presentan la frecuencia de los ejemplos asociados a cada clase.

Figura 4-12: Árbol y regla de aprendizaje para el criterio de reactividad del servicio buyItem

```
A

Árbol

1   buyItem(-A,-B)
2   hour17(A) ?
3         +-yes:clientPossesitem12303(A) ?
4         |       +-yes: time200(A) ?
5         |       |      +-yes: thursday(A) ?
6         |       |      |      +-yes: [failure] 12.0 [[success:2.0,failure:10.0]]
7         |       |      |      +-no:  [success] 5.0 [[success:3.0,failure:2.0]]

Regla

buyItem(A,[success:0.166666666666667,failure:0.833333333333333]) :- hour17(A),clientPossesitem12303(A),time200(A),thursday(A), !.
% 10.0/12.0=0.833333333333333
```

En la parte B de la figura 4-12, se muestra la regla de aprendizaje generada a partir del árbol descrito en la parte A. Esta regla contiene el nombre del servicio ejecutado y los valores particulares de cada clase y describe el camino del árbol es decir, los nodos hoja que le permiten alcanzar dichos valores para las clases. Esta regla es una representación prolog del árbol de aprendizaje.

El conjunto de reglas R emitidas por el aprendizaje de máquinas, son traducidas y almacenadas en la base de datos knowledgeDB para ser utilizadas en posteriores composiciones. Estas reglas, representan el riesgo de ejecutar un servicio con un límite de tiempo de ejecución en periodos de días y horas. El algoritmo de traducción y almacenamiento en la base de datos se describe en la figura 4-13.

Figura 4-13_ Algoritmo de traducción y almacenamiento de reglas de aprendizaje para el criterio de reactividad

1. **while** $\exists\, r_j \in R\,(s_i)$ **do**
2. $\quad \forall\, r \exists\, d \cup h \cup te \mid d \in dia\ h \in hora\ te \in valor\ de\ restricción\ de\ tiempo$
3. $\quad r_j.value = número$
4. $\quad$ **if** (r_j)**then**
5. $\quad\quad knowledgeDB \leftarrow r_j$
6. $\quad\quad performanceValue(s_i) \leftarrow r_j.value$
7. $\quad$ **endif**
8. **endwhile**

Por cada regla generada para un servicio Web si, ésta debe contener un día d, una hora h, una restricción de tiempo te y un objetivo goal, esta validación se referencia en la línea

2 del algoritmo, el valor numérico de la regla se almacena de acuerdo a la línea 3. Si existe una regla con las condiciones expresadas en la línea 2, la regla es creada en la base de datos (línea 4) y finalmente se asocia el valor de riesgo que expresa la regla en la base de datos (línea 5).

4.4 Módulo de manejo de creencias

Este módulo, requiere las observaciones de la instanciación de entradas y salidas de cada servicio ejecutado, con el fin de generar las reglas de aprendizaje que identifican cuál de ellas tienen que ser prioritarias en la instanciación de los servicios.

4.4.1 Adquisición de información de manejo de creencias relacionadas con los servicios Web

Por cada ejecución de un servicio s_i, se captura el conjunto de entradas que ese servicio requiere para ejecutarse y las salidas que produce ese conjunto de entradas, las cuales son capturadas en una observación O_j a través de la tupla $(s_j, inputs, outputs, cl)$ donde:

s_j: representa el servicio que se ejecutó.
$inputs$: son las entradas que el compositor asume para el servicio cuando lo envía a ejecución.
$ouputs$: representa las salidas del servicio que el compositor asume como verdaderas cuando envía al servicio a ejecutarse.
cl: representa la clase de relación que existe entre las salidas supuestas por el sistema de composición y las salidas emitidas por el servicio después de su ejecución. Las clases son: éxito cuando existe un emparejamiento entre las salidas supuestas por el compositor y las salidas reales adquiridas del mundo real y fallo, cuando no hay dicho emparejamiento.

La figura 4-14, se presenta un ejemplo de las observaciones del manejo de creencias.

Figura 4-14: Observación de manejo de creencias del servicio buyItem

```
Observaciones de la ejecución del servicio buyItem
Criterio Manejo de Creencias

buyItem,&Item12303,$TV,$LG,client_possess(item12303),sucess.
buyItem,&Item12302,$Mp3,$Sony,client_possess(item12303),failure.
buyItem,&Item12301,$Laptop,$hp,client_possess(item12303),sucess.
buyItem,&item12301,$iphone,$apple,client_possess(item12303),failure.
buyItem,&item12303,$TV,$LG,client_possess(item12303),failure.
```

Esta figura, presenta un conjunto de cinco observaciones O_j, cada una está compuesta de cuatro aspectos particulares: el nombre del servicio: buyItem, el conjunto de entradas identificadas a través del símbolo &, el conjunto de salidas identificado por el símbolo $. Estos símbolos, se anteponen a los datos como por ejemplo: &Item12303 para la entrada

de la primera observación y $TV y $LG las salidas de dicha observación. El último aspecto de las observaciones es la clase success o failure según la evaluación de las creencias del compositor con el mundo real.

Una vez adquirida esta información, se traslada a la representación del lenguaje lógico a través de un algoritmo el cual obtiene la información del conjunto de observaciones O_j por cada servicio. El algoritmo y el lenguaje lógico para el criterio de manejo de creencias se presentan en la figura 4-15. En el lenguaje lógico se describe la función objetivo en la parte (A) y la base de conocimiento en la parte (B).

Figura 4-15: Algoritmo y ejemplo del lenguaje lógico del criterio manejo de creencias del servicio buyItem

Algoritmo	Lenguaje Lógico
1. ***while*** ∃ O_j s_i ***do*** 2. $\tau \leftarrow s_i \cup cl$ 3. $kb \leftarrow inputs \cup outputs$ 4. ***end while***	A % Función objetivo execution_buyItem(o,class). classes([success,failure]). B % Ejemplo o1 %Instanciación función objetivo execution_buyItem(o1,success). %Base de conocimientos input_item12303(o1). output_TV(o1). output_LG(o1). client_possess{item12303}(o1). % Ejemplo o2 %Instanciación función objetivo execution_buyItem(o2,failure). %Base de conocimientos input_item12303(o2). output_MP3(o2). output_sony(o2). client_possess{item12303}(o2).

El algoritmo de la figura 4-15, consiste en capturar la información de las observaciones de cada servicio. En primer lugar se asocia a la función objetivo τ el servicio s_i y la clase de ejecución de éste es decir éxito o falla (línea 2) y posteriormente adicionar las entradas y las salidas del servicio a una representación de la base de conocimientos kb (línea 3).

La parte A en el lenguaje lógico, presenta la estructura de la función de objetivo, en ésta se observa la utilización de la palabra execution_ seguida del nombre del servicio ejecutado. Esta función toma una constante o la cual sirve como un identificador utilizado por el sistema Tilde para enlazar la función objetivo con la base de conocimientos. Las clases representan la variación de la función objetivo. En la parte B se observa la instanciación de la función objetivo y la construcción de la base de conocimientos. En esta estructura, se antepone la palabra input_ con el fin de representar las entradas del servicio y output_ para representar las salidas del servicio. Para enlazar estos datos pertenecientes a una misma observación se utiliza el identificador o seguido de un número consecutivo.

4.4.2 Generación del conocimiento de control para el manejo de creencias

Este criterio, es aprendido a través de árboles de decisión lógicos. Por cada servicio s_i, se genera un árbol de aprendizaje, el cual ajusta las ejecuciones del servicio en función de la relación entre entradas y salidas de dichos servicios, determinando cuando son éxito y cuando falla. En el árbol generado, cada una de las ramas produce una regla $r_j \in R$ de aprendizaje del comportamiento del servicio s_i

En la figura 4-16 se muestra un ejemplo del árbol de decisión lógico y las reglas generadas para el servicio buyItem.

Figura 4-16: Árbol y regla de aprendizaje para el criterio manejo de creencias del servicio buyItem.

```
A

        Arbol

        buyItem(-A,-B)
        output_TV(A) ?
        +--yes: [success] 15.0 [[success:15.0,failure:10.0]]

B
        Regla

        buyItem(A,B,[success:0.6,failure:0.0]) :- output_TV(A), !.
        % 15.0/25.0=0.6
```

En la parte A de esta figura se presenta el árbol de aprendizaje generado por el sistema Tilde. Aquí se observa que solo se generó un nodo hoja con la salida TV y describe su valor de éxito. Este valor representa la frecuencia de los ejemplos encontrados bajo esa característica.

La parte B, presenta la regla de aprendizaje para dicho árbol. En esta se observa que al ejecutar el servicio buyItem, implica que existe una probabilidad del 60% que el servicio tenga como salida TV.

Este criterio, proporciona directamente a través de las reglas aprendidas r_j, el conjunto de instancias más representativas de cada servicio s_i teniendo en cuenta el dominio del problema, con el objeto de ser asignadas al sistema de composición. Dichas reglas son almacenadas en la base de datos knowledgeDB para ser usadas durante la instanciación de cada servicio, de manera que sean especificadas de acuerdo a su probabilidad de éxito en la ejecución del servicio.

La figura 4-17, presenta el algoritmo para traducir y asignar al sistema de composición las reglas de aprendizaje generadas.

Figura 4-17: Algoritmo de traducción y almacenamiento de reglas de aprendizaje para la criterio de manejo de creencias.

```
1. while r_j ∈ R(s_i) do
2.     if ∃ domain then
3.         if (inputs ∪ outputs) ≡ (inputs ∪ outputs') then
4.             if (r_j) then
5.                 knowledgeDB ← r_j
6.             end if
7.         end if
8.     end if
9. end while
```

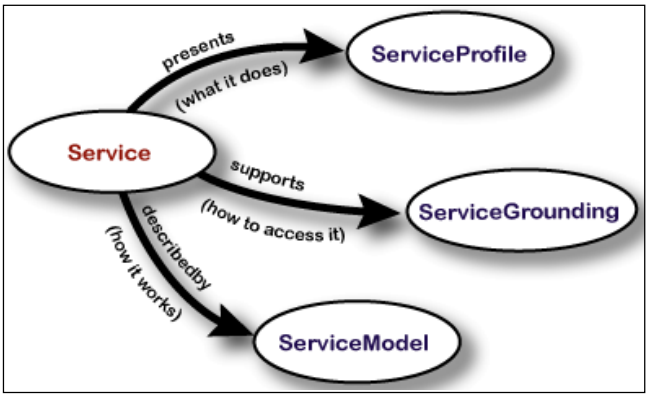
La línea 3, permite comparar las entradas y salidas (inputs, outputs) que el compositor de servicios sugirió como buenas antes que el servicio se ejecutara con las entradas y salidas (inputs', outputs') reales del servicio. Si tanto las salidas como las entradas son iguales, se pregunta si existe una regla que cumpla la condición anterior, ésta es creada en la base de datos (línea 5).

4.5 Módulo de integración del modelo con la planificación

Este módulo, permite asociar la información obtenida a partir de las reglas de aprendizaje, con el modelo de planificación. Para esto, el módulo de integración, recibe como entradas el dominio de composición sobre el cual se construirá el plan, los servicios participantes en la composición y los valores de riesgo de disponibilidad y reactividad aprendidos por el modelo de aprendizaje para finalmente, construir como salida una nueva representación semántica de los servicios a través de una ontología OWL-S integrada con una ontología QoS.

Para esta integración, se propone hacer uso de la ontología OWL-S para capturar la descripción semántica de los servicios. Esta descripción OWL-S está orientada a tres dimensiones como se muestra en la figura 4-18

Figura 4-18: Ontología OWL-S [(Martin, y otros, 2004)].



- Perfil del servicio (ServiceProfile): describe las funcionalidades de un servicio en términos de entradas, salidas, precondiciones y efectos.
- Modelo del servicio (ServiceModel): describe como usar un servicio, detallando sus parámetros e indicando paso a paso el proceso que conduce a los resultados de estos parámetros.
- ServiceGrounding: describe como tener acceso a un servicio. En particular, esto especifica un protocolo de comunicación y formatos de mensaje.

Si bien es cierto, en esta descripción se cubren varios de los aspectos representativos de los servicios, no cuenta con una parte particular para definir las características de fallos de un servicio. No obstante, el perfil del servicio se utiliza para especificar las características de fallos a través de sus parámetros de calidad (QoS). Para esto, se hace referencia a los parámetros del servicio (serviceParameter), el cual permite describir parámetros adicionales que los servicios Web requieren y que no son parte funcional de este.

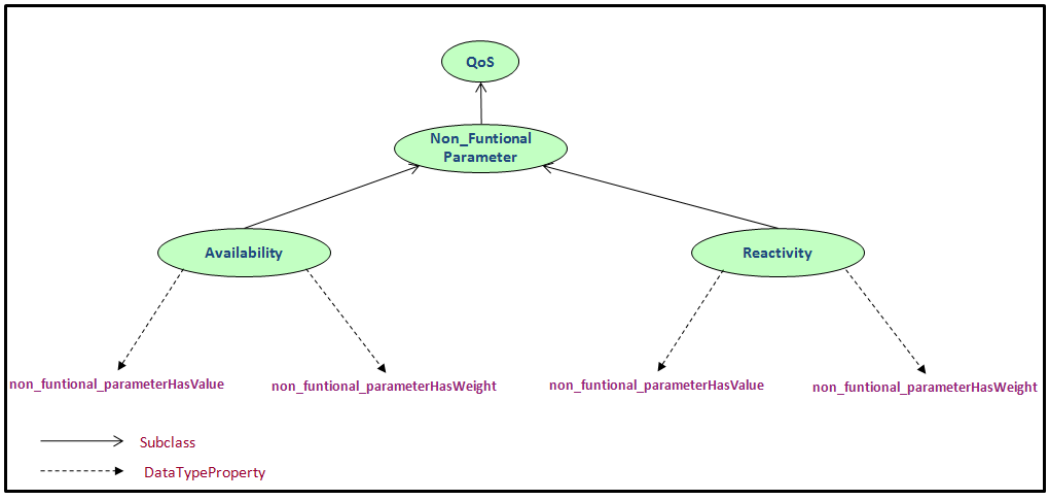
4.5.1 Modelo de la ontología QoS

La ontología de calidad QoS que se propone en esta tesis, se encuentra enlazada al perfil del servicio a través del serviceParameter el cual, representa una ontología que especifica elementos adicionales que describen las características particulares de un servicio Web que no se encuentran representadas en la ontología OWL-S.

Esta ontología, se extiende del serviceParameter a través de una subclase denominada QoS. Esta clase, es la principal de la ontología de calidad. De esta clase, se deriva una subclase denominada non_Funtional_Parameter la cual, contiene una subclase por cada criterio de calidad que representa el riesgo de falla. Por lo tanto, esta ontología es capaz de soportar cualquier parámetro de calidad a partir de tres valores particulares: nombre, valor e impacto del parámetro. Es decir, cada vez que se quiera adicionar un QoS a un servicio Web, se crea una instancia de esta subclase. Cada instancia de una subclase de este tipo, contiene dos propiedades de tipo real, las que permiten especificar el valor del QoS y su respectivo impacto.

Para el caso de esta tesis, se crearon dos subclases y sus instancias que representan los criterios de disponibilidad y reactividad de los servicios propuestos en nuestro modelo para el manejo de riesgos. Sus valores son extraídos a partir del conocimiento de control almacenado en la base de datos, y sus impactos se asocian dependiendo de la evaluación que se requiera es decir, aplicando la ecuación 2.8 descrita en el capítulo 2. La figura 4-19. Presenta la ontología de calidad para cada servicio.

Figura 4-19: Ontología de calidad.



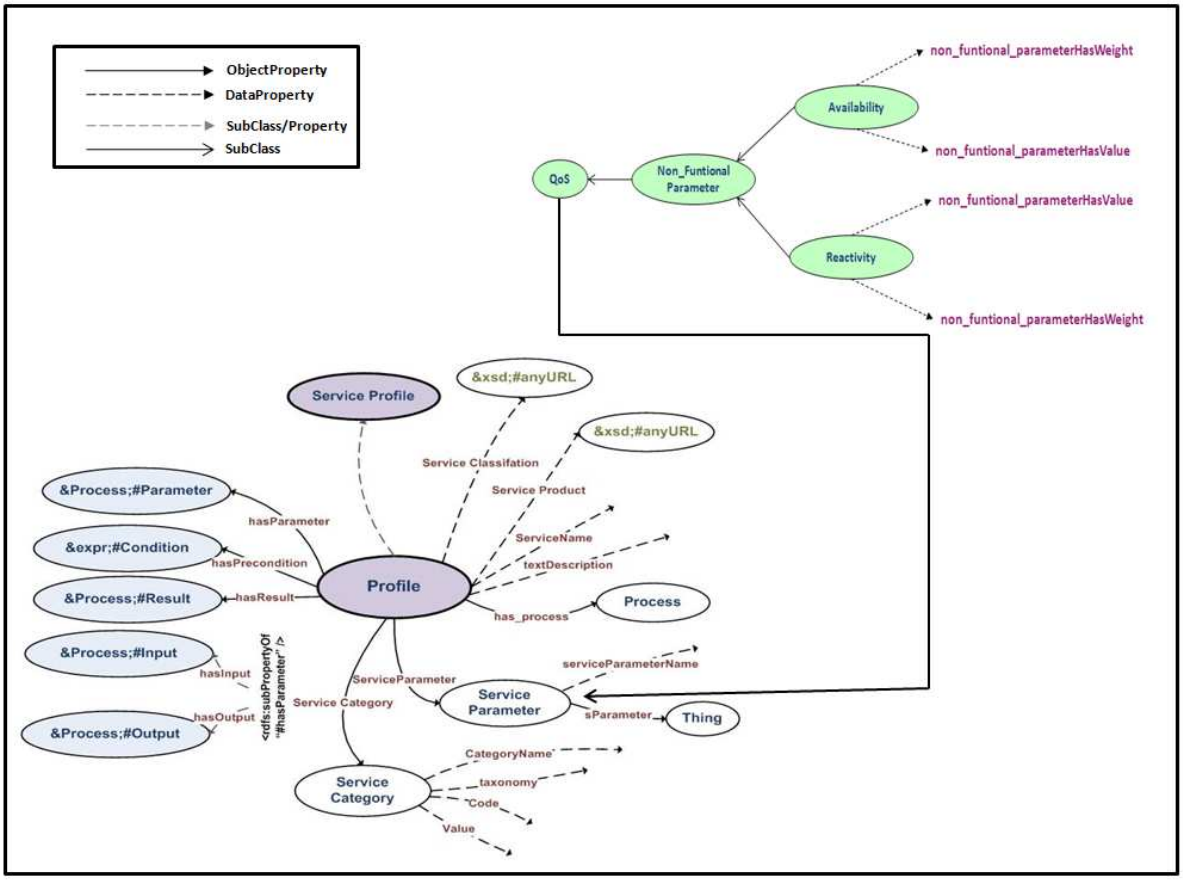
En el anexo B, se presenta la ontología QoS descrita en el lenguaje owl.

4.5.2 Integración de datos

Para la integración de los datos del modelo de aprendizaje con el modelo de planificación se propone un segundo traductor, el cual se encarga de capturar la información asociada a los criterios $c = \{\text{disponibilidad y reactividad}\}$ para cada servicio s_i que participa en la composición y, construye una nueva descripción semántica para dichos servicios newOnto, reemplazando la descripción ontológica de cada s_i. de tal manera que un servicio Web está representado por una ontología extendida de calidad como se presenta en la figura 4-20.

En el anexo C, se muestra la representación ontológica del perfil del servicio buyItem, servicio perteneciente al dominio de compras por internet.

Figura 4-20: Extensión de la ontología profile con soporte para QoS



4.6 Discusión

En este capítulo, se presentó el modelo de aprendizaje para el manejo de riesgos de falla en composiciones de servicios Web, utilizando tres criterios de calidad: manejo de creencias, disponibilidad y reactividad de los servicios; los cuales, evalúan las ejecuciones de los servicios y permiten determinar el factor de riesgo para cada servicio Web. Aunque estos criterios evalúan independientemente a los servicios, éstos participan en el plan total de composición, evaluando así su comportamiento en conjunto para encontrar el valor final del riesgo del servicio.

El modelo, está constituido por un módulo de aprendizaje que es el encargado de generar el conocimiento de control que evalúa el comportamiento de los servicios Web, a partir de sus ejecuciones y el módulo de integración es el encargado de hacer la comunicación entre el módulo de aprendizaje y un sistema de planificación. Este sistema de planificación, se enfoca a la composición de servicios Web, el cual trata de encontrar el conjunto de servicios Web que a partir de unas condiciones iniciales, alcanzan un objetivo particular.

Al trabajar con servicios Web semánticos, existe la necesidad de tener una descripción que haga posible identificar las características de los servicios de manera semántica; una de esas representaciones es a partir de ontologías. Es así como, se presentó una extensión a la ontología serviceParameter para hacer la representación de QoS dentro

de la ontología OWL-S. Esta ontología, evalúa cualquier tipo de criterio de calidad siempre que cuente con un nombre, un valor y un peso.

5. DOMINIUS-BP: Sistema de Adquisición de Conocimiento de Control para Manejo de Fallas en Composición de Servicios

En este capítulo, se representa la aplicación de software que se ha implementado para evaluar el modelo de aprendizaje propuesto en el capítulo 4 de esta tesis el cual, permite adquirir de manera automática el conocimiento de control necesario para manejar el riesgo de fallas en composiciones de servicios.

5.1 Arquitectura del sistema Dominius-BP

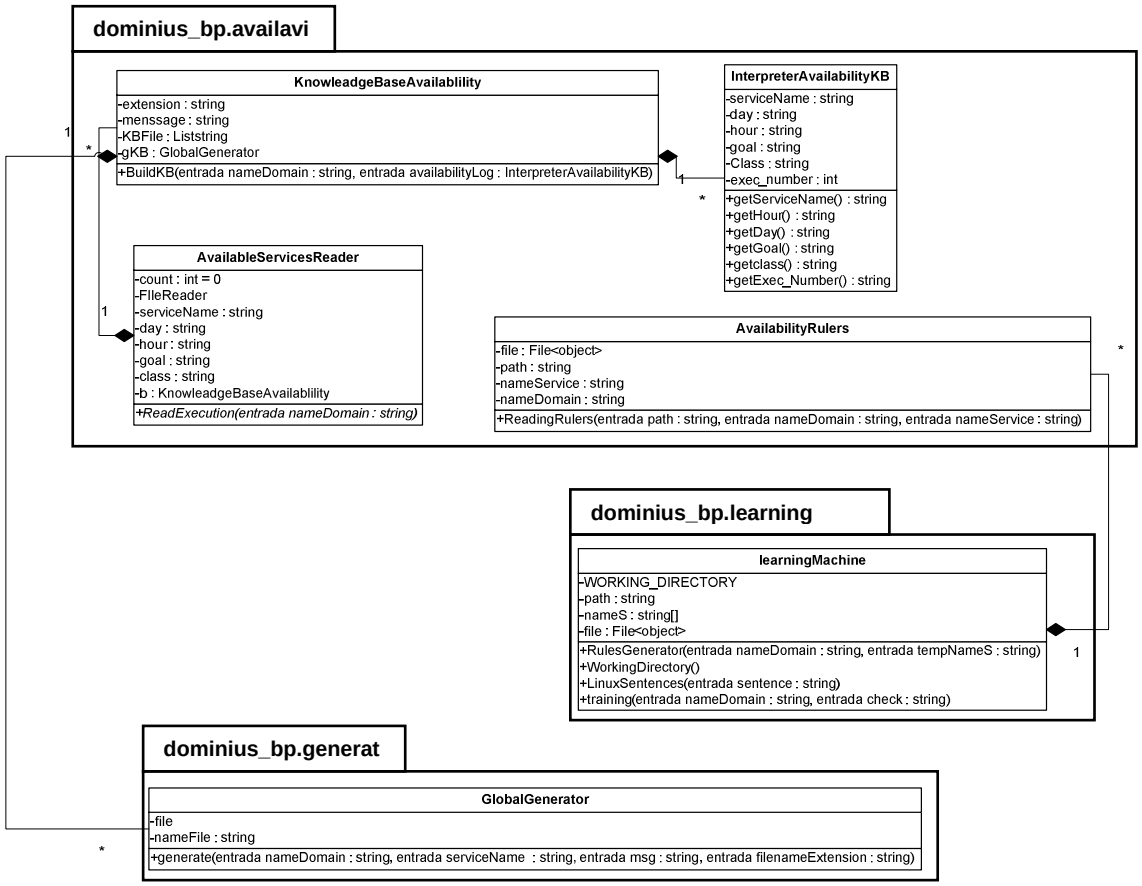
El objetivo principal de esta arquitectura, es la adquisición del conocimiento de control para el manejo de fallas en las composiciones de servicios. Para esto, esta arquitectura implementa cuatro componentes principales. Haciendo una similitud al modelo teórico, tres componentes están relacionados directamente con cada criterio de falla, esto haciendo referencia a los criterios de disponibilidad, reactividad y manejo de creencias de los servicios Web. El cuarto componente, es el integrador del sistema de planificación orientado a la composición de servicios. Para modo de pruebas se utilizó el sistema INDYGO con el sistema de aprendizaje DOMINIUS-BP. Adicionalmente, se cuenta con un paquete de interfaz gráfica el cual permite al usuario interactuar con el modelo de aprendizaje.

5.2 Diseño e implementación del componente de disponibilidad

En este componente, se pretende adquirir aquella información relacionada con la disponibilidad de los servicios durante su ejecución, y trasladarla a un lenguaje de aprendizaje de máquinas para posteriormente obtener el conocimiento de control para cada servicio ejecutado.

El diagrama de clases del componente de disponibilidad se presenta en la figura 5-1.

Figura 5-1: Diagrama de clases del componente de disponibilidad



A continuación se describen las clases principales utilizadas para este componente.

5.2.1 Clases para la adquisición y traducción de especificaciones de disponibilidad de los servicios (AvailableServicesReader, KnowleadgeBaseAvailablility)

Existen dos clases encargadas de realizar la adquisición y la traducción de las especificaciones de disponibilidad de los servicios. En primer lugar, la clase AvailableServicesReader es la encargada de leer los datos de la ejecución de los servicios a través del método readExecution, y almacenar estos datos en la estructura InterpreterAvailabilityKB. Esta estructura se encuentra constituida por: el nombre del servicio ejecutado, el día y la hora de ejecución, la clase de ejecución que tuvo el servicio (si fue exitosa o fallida), el objetivo del plan de composición al que pertenece el servicio y un consecutivo para identificar el número total de ejecuciones.

Una vez almacenados todos los datos de disponibilidad asociados a la ejecución de cada servicio, entra en operación la segunda clase de este componente: KnowleadgeBaseAvailablility la cual, es la encargada de interpretar la información de la estructura InterpreterAvailabilityKB a través del método BuildKB, y crea un archivo que

representa la función objetivo y el conjunto de ejemplos entrenados producto de las ejecuciones de cada servicio. Este es un proceso que se repite por cada uno de los servicios ejecutados.

5.2.2 Clase para la generación del conocimiento de control de disponibilidad de los servicios (LearningMachine, AvailabilityRulers)

Para generar la información de control respecto a la disponibilidad de los servicios, se utiliza la clase LearningMachine, esta clase, genera la configuración de la herramienta TILDE a través del método RulesGenerator y hace un llamado a esta herramienta por medio del método LinuxSentences para construir a partir de los archivos generados por las clases anteriores, un archivo que representa el conjunto de reglas de aprendizaje, es decir el conocimiento de control para cada servicio. Finalmente, el componente AvailabilityRulers a través del método ReadingRulers es el encargado de traducir esas reglas y almacenarlas en la base de datos knowledgeDB.

5.3 Componente de reactividad de servicios

5.3.1 Clases para la adquisición y traducción de especificaciones de reactividad de los servicios (ReactivityServicesReader, KnowleadgeBaseReactivity)

Al igual que en el componente anterior, los datos de ejecución son emitidos por el sistema de composición. Para leer estos datos, se utiliza el método ReadExecution de la clase performanceServicesReader; y esta misma clase almacena estos datos en la estructura InterpreterReactivityKB. La estructura InterpreterReactivityKB, está compuesta por: el nombre de servicio ejecutado, el tiempo, el día y la hora de ejecución, la clasificación de la ejecución de éxito o fallo dependiendo de la restricción de tiempo establecida, el objetivo del plan al que pertenece el servicio y un consecutivo para identificar el número de ejecuciones totales. Cuando el conjunto de datos de ejecución de los servicios referentes a la reactividad están almacenados, la clase KnowleadgeBaseReactivity a través del método BuildKB, construye un archivo que contiene el conjunto de ejemplos entrenados por cada servicio y la función objetivo de acuerdo a la reactividad de los servicio.

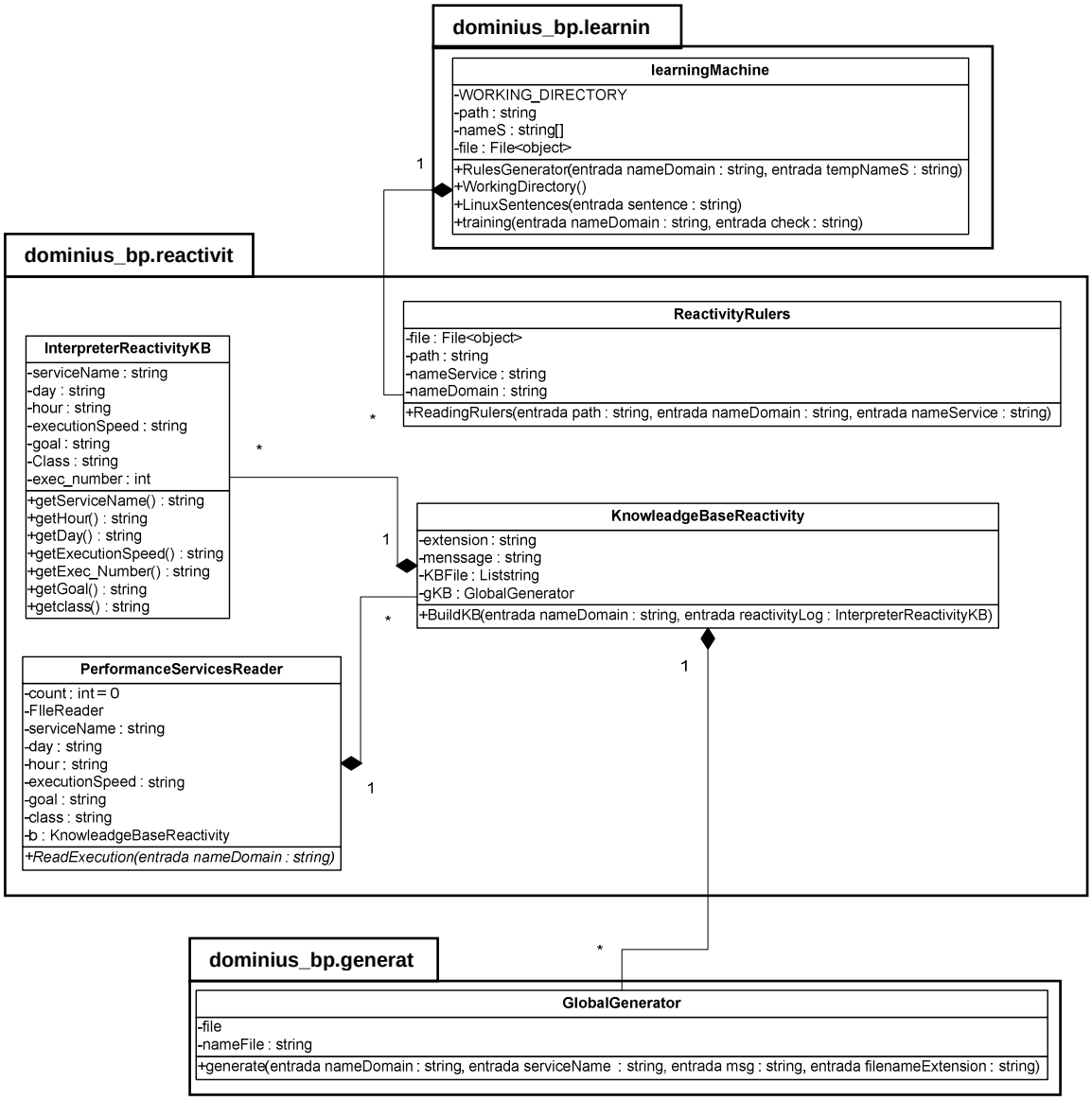
5.3.2 Clases para la generación del conocimiento de control de reactividad de los servicios (LearningMachine, ReactivityRulers)

Aquí, también se hace uso de la clase LearningMachine para ejecutar la herramienta de aprendizaje TILDE, la cual toma los archivos del conjunto de ejemplos entrenados y genera un documento con la información de aprendizaje. Este componente, ejecuta una clasificación utilizando los árboles de decisión y genera las reglas de aprendizaje lógicas por cada servicio ejecutado.

Una vez encontradas las reglas de aprendizaje, la clase ReactivityRulers a través del método ReadingRulers, se encarga de interpretar cada una de esas reglas y trasladarlas a la base de datos knowleageDB.

El diagrama de clases del componente de Reactividad se muestra en la figura 5-2.

Figura 5-2: Diagrama de clases del componente de reactividad

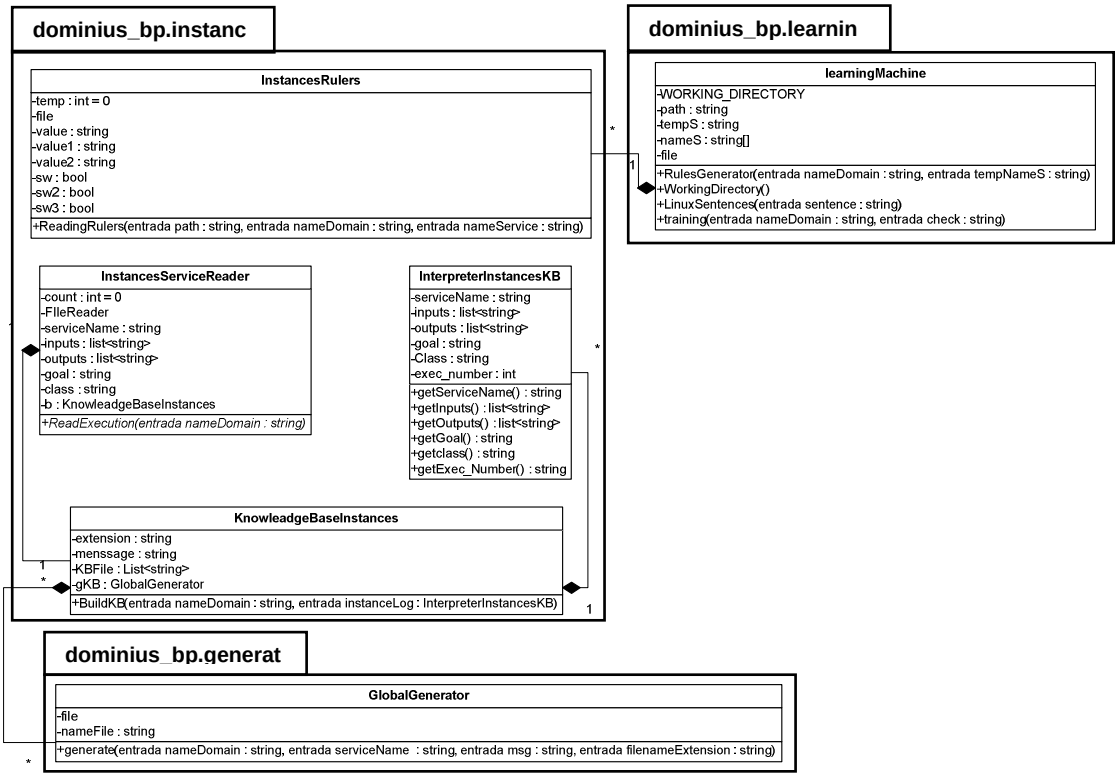


5.4 Componente de manejo de creencias

5.4.1 Clases para la adquisición y traducción de especificaciones de manejo de creencias del compositor de servicios (InstancesServicesReader, KnowleadgeBaseInstances)

La figura 5-3**:** muestra el diagrama de clases del componente manejo de creencias.

Figura 5-3: Diagrama de clases del componente de manejo de creencias



En el manejo de creencias, la clase utilizada para la adquisición de la información es InstancesServicesReader. Esta clase, toma cada uno de los datos que representan la ejecución de un servicio y los traslada a la estructura InterpreterInstancesKB, la que almacena el nombre del servicio ejecutado, sus entradas y salidas, la clasificación si fue exitosa o no y el objetivo de la composición. Esta información solo es adquirida para servicios que adquieran información del mundo y no lo cambien.

Posterior a este almacenamiento, la segunda clase KnowleadgeBaseInstances traduce una por una la información de cada servicio a una representación de ejemplos entrenados en los cuales, se describe la función objetivo y la base de conocimiento que representa la ejecución de cada uno de los servicios.

5.4.2 Clases para la generación del conocimiento de control de manejo de creencias del compositor de servicios (LearningMachine, InstancesRulers)

En cuanto a la generación del conocimiento de control, la clase LearningMachine es utilizada para ejecutar remotamente la máquina de aprendizaje TILDE, Esta herramienta, genera un archivo de reglas lógicas con la información de las entradas y salidas que requiere un servicio para ejecutarse. Finalmente estas reglas se traducen a la base de datos knowleageDB haciendo uso de la clase InstancesRulers a través del método ReadingRulers, para ser utilizada posteriormente.

5.5 Componente de integración

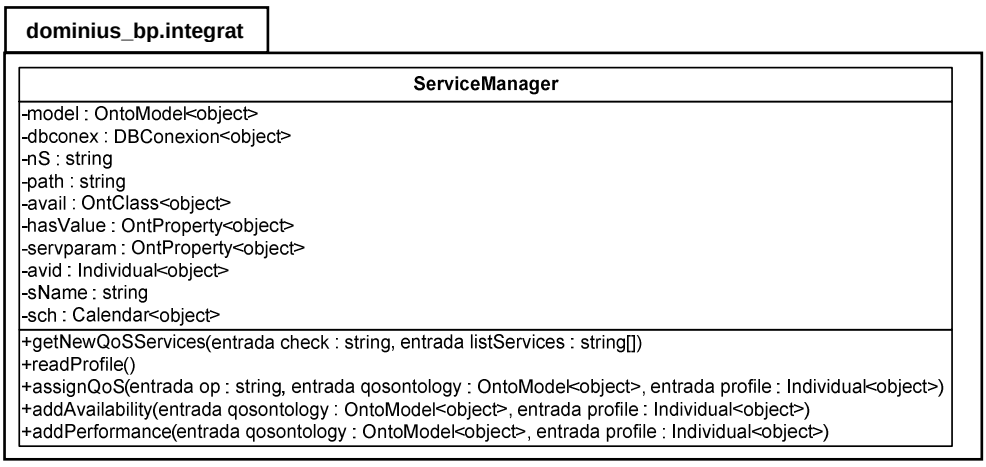
Este componente a través de la clase (serviceManager), lee la lista de servicios que se utilizan para una composición y crea por cada servicio una representación ontológica del mismo, adicionándole los criterios que definen el riesgo de falla ocasionados por el entorno del servicio.

Para esto, esta clase toma una versión de un perfil del servicio (profile) y le adiciona la ontología de calidad con los valores de riesgo de falla para cada criterio. A partir de esto, crea una nueva ontología por cada servicio y construye una nueva lista de servicios.

Para identificar cuáles son los valores de cada criterio para el servicio a ejecutar, hace una conexión a la base de datos knowledgeDB y abstrae por cada servicio la información del criterio requerido asociado a dicho servicio.

Este módulo, es funcional para los criterios de disponibilidad y reactividad de los servicios. La descripción gráfica de la clase ServiceManeger se muestra a continuación en la figura 5-4.

Figura 5-4: Clase ServiceManager



5.6 Componente de interfaz gráfica (gui).

Este componente consta de 4 componentes gráficos descritos a continuación.

5.6.1 Interfaz principal DominiusBPGUI

Esta interfaz, es la primera que se presenta al usuario. Esta ventana, proporciona la visual de tres posibles usos que implementa el sistema DOMINIUS-BP:

- Generación de ejemplos entrenados,
- Generación de los archivos de aprendizaje y conocimiento de control y
- Integración del sistema de planificación con el sistema de aprendizaje

Cada uno de estos tres usos representa las tres interfaces restantes de este componente. La figura 5-5 presenta la interfaz principal del sistema DOMINIUS-BP.

Figura 5-5: Interfaz gráfica principal

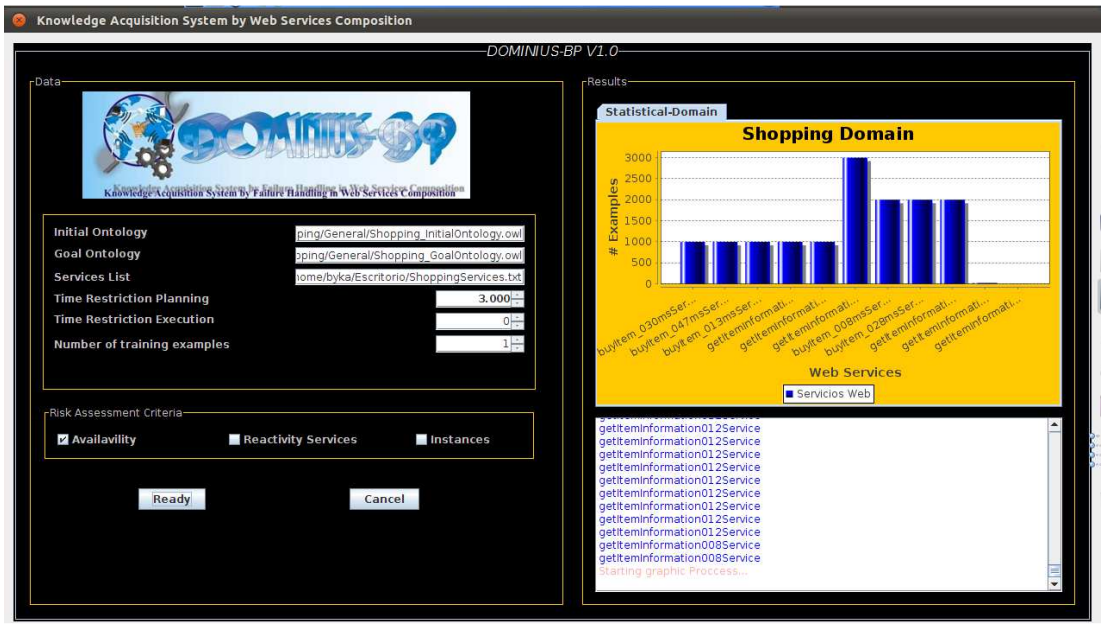
5.6.2 Interfaz de Generación de Ejemplos de Entrenamiento e Integración LearningDataGUI

Esta interfaz, cumple dos funciones: en primer lugar, es utilizada para hacer un llamado al sistema de planificación orientado a la composición de servicios y generar el conjunto de ejemplos de entrenamiento para el sistema de aprendizaje DOMINIUS-BP. Este componente, hace uso del sistema de composición INDYGO por lo tanto, requiere como entrada las siguientes especificaciones: Una ontología inicial descrita con especificaciones owl, una ontología final igualmente descrita en owl, una lista de servicios y una restricción de tiempo de planificación. Adicionalmente, este componente requiere:

el número de ejemplos de entrenamiento a generar, el criterio o criterios de fallo que se desea capturar en la ejecución de los servicios y, si uno de los criterios es la reactividad de los servicios, se hace necesario ingresar al sistema la restricción de tiempo de ejecución para los servicios.

Al finalizar este proceso, esta interfaz muestra al usuario una gráfica con el conjunto total de servicios utilizados para alcanzar el plan tras varias ejecuciones. La figura 5-6 presenta la interfaz para generar ejemplos de entrenamiento.

Figura 5-6: Interfaz gráfica para generación de ejemplos de entrenamiento



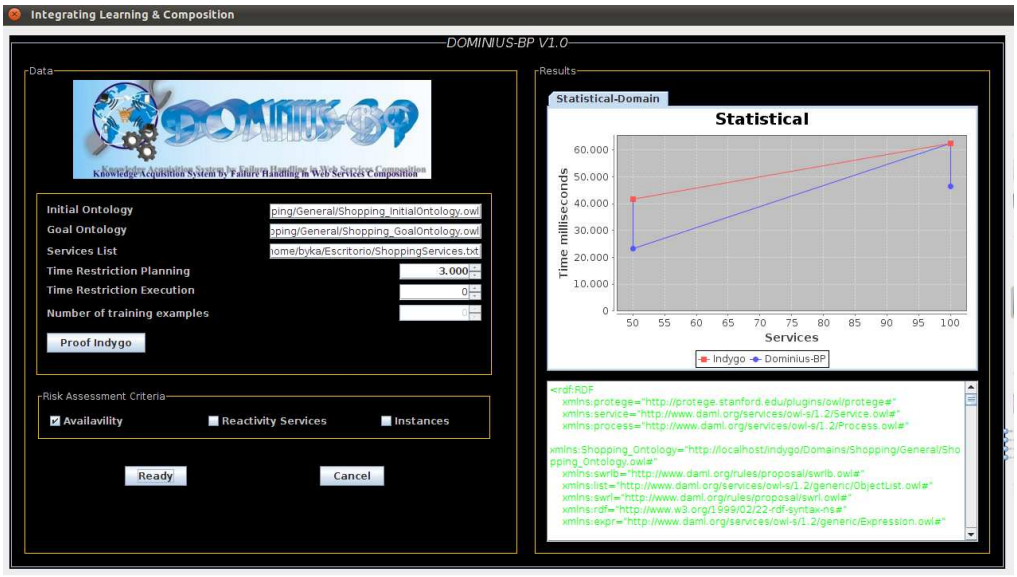
En segundo lugar, este componente también se utiliza para la integración del sistema de aprendizaje con el sistema de composición. A diferencia de lo anterior, desaparece la evaluación del número de ejemplos entrenados requerido y aparece una nueva acción que es la utilización del sistema de composición para efectos de comparación sin el sistema de aprendizaje y con éste.

Al finalizar el proceso, esta interfaz presenta una gráfica comparativa entre el sistema de composición utilizando el sistema de aprendizaje y sin utilizarlo. La figura 6.7 muestra la interfaz gráfica utilizada para la integración del modelo de planificación orientado a la composición y el modelo de aprendizaje.

5.6.3 Interfaz de Aprendizaje TrainingDataLearningGUI

Es una interfaz gráfica que le permite seleccionar al usuario el dominio al cual se le generarán los archivos de aprendizaje teniendo en cuenta el riesgo que se desea medir, es decir escoger entre los criterios que evalúan el riesgo o riesgos del servicio.

Figura 5-7: Interfaz gráfica de integración



Como salida, este componente presenta al usuario el conjunto de ejemplos de entrenamiento, el árbol de aprendizaje generado y sus reglas de aprendizaje asociadas. La figura 5-8, presenta esta interfaz.

Figura 5-8: Interfaz Gráfica de Aprendizaje

5.7 Discusión

En este capítulo, se describe la aplicación de software implementada en función del modelo teórico descrito en el capítulo 4 y apoyado en los capítulos 2 y 3 de esta tesis. De esta aplicación resulta el prototipo DOMINIUS-BP el cual, permite adquirir de forma automática el factor de riesgo de falla en composiciones de servicios.

DOMINIUS-BP, define aspectos básicos del aprendizaje para utilizarlos en composiciones de servicios.

- Obtiene el conjunto ejecuciones de cada servicio participante en un plan de composición para convertirlo en un ejemplo de entrenamiento que alimenta la base de conocimientos del aprendizaje de máquinas.
- Construye el ambiente de aprendizaje para un conjunto de servicios teniendo la información de su ejecución como lo es el día y la hora de ejecución, el plan en el que se utilizó y las restricciones de tiempo de ejecución de ser necesarias. Asimismo, construye un ambiente particular de aprendizaje a partir del comportamiento del sistema de composición identificando las creencias asumidas por este entorno a las entradas y salidas del servicio.
- Traslada una descripción ontológica tradicional de un servicio a una descripción ontológica basada en fallas para mejorar la composición y disminuir el riesgo de fallas en la misma.
- Permite hacer una comparación entre el sistema de composición haciendo uso del sistema de aprendizaje y, la ejecución del sistema de composición sin apoyo del aprendizaje, en el cual se hace una comparación gráfica de riesgos en la composición de servicios.

6. Evaluación y experimentación

En este capítulo se presenta la validación del modelo de aprendizaje propuesto en el capítulo 5 utilizando el prototipo DOMINIUS-BP y describiendo un ambiente de pruebas en el que se considera los criterios definidos en el capítulo 2.

6.1 Características del ambiente de pruebas

Las características técnicas utilizadas para el ambiente de pruebas del modelo de aprendizaje fueron:

- Computador con procesador Intel Core i3 (2.27GHZ), DDR 3799 con sistema operativo Ubuntu 10.10 kernel 2.6.35-28-generic 64 bits en el cual se implemento el prototipo DOMINIUS-BP.
- Sistema de planificación orientado a la composición de servicios INDYGO (Guzman & Ovalle, 2009), (Guzman & Ovalle, 2008), (Guzman & Ovalle, 2008b).
- Sistema de aprendizaje de máquinas TILDE (Blockeel & Raedt, 1998) para la generación de árboles lógicos

El dominio de composición que se implementó para este ambiente de pruebas fue el dominio de compras por internet. Este dominio, consiste en un modelo de compras donde un usuario utiliza la Web para obtener información de algún artículo que desea comprar y la forma de adquirir dicho artículo. Para este dominio, se implementaron 10 servicios Web los cuales representan las acciones que son requeridas al instante de interactuar en una compra por la Web.

Los servicios Web se separaron de la siguiente forma: 5 servicios para obtener información de artículos identificados como: getItemInformation001, getItemInformation002, getItemInformation003, getItemInformation004, getItemInformation005 y 5 servicios de comprar artículos identificados como: buyItem001, buyItem002, buyItem003, buyItem004, buyItem005.

Los servicios de obtener información reciben una entrada que hace referencia al código del artículo a comprar y genera cuatro salidas: titulo, precio, descripción del artículo y la confirmación SW si el artículo se encuentra disponible. Por su lado, los servicios para comprar el artículo, reciben como entradas: el código del artículo, un número de tarjeta de crédito y la fecha de caducidad de la misma. Como resultado, la propiedad del artículo es transferida al cliente.

Este conjunto de 10 servicios permite evaluar los tres criterios de riesgos descritos en el capítulo 3 de esta tesis. La importancia de los servicios de getItemInformation radica en la particularidad de medir el criterio de manejo de creencias por parte del compositor puesto que estos servicios sirven como servicios de consulta en los cuales se averigua la información del mundo.

6.2 Precisión del modelo Dominius-BP

La precisión del modelo de aprendizaje planteado en este trabajo de tesis, está asociada directamente con los resultados proyectados por el aprendizaje propuesto y la certeza de las reglas generadas por el mismo en cuanto al comportamiento de los servicios a través del tiempo.

Para el modelo de aprendizaje es fundamental la regularidad en el comportamiento del servicio en cuanto a sus métricas de disponibilidad y reactividad, esto permite que el conocimiento de control que el modelo de aprendizaje genere sobre dichas variables de cada servicio para cada día y hora definidos se aproxime más al comportamiento real de los mismos.

Asimismo el número de ejemplos de entrenamiento influye directamente en el nivel de aprendizaje adquirido y por lo tanto en el grado de certeza de las reglas generadas por el aprendizaje en cuanto al comportamiento de los servicios a través de los distintos días y horas. Así, en cuanto mayor sea el número de ejemplos de entrenamiento utilizados mejor será la respuesta del sistema de aprendizaje.

El experimento consiste entonces en evaluar el modelo del comportamiento de los servicios aprendido por el sistema Dominius-BP. El experimento muestra como el error del modelo de aprendizaje varía según el número de ejemplos de aprendizaje de manera inversamente proporcional, enfocándose en la exploración del ambiente.

Para la estimación del error, se utilizó el cálculo del error cuadrático medio (Mood, Graybill, & Boes, 1974). Este estima el error de un estadístico a partir del cuadrado de las medias aritméticas respecto al valor verdadero del estadístico que se trata de estimar, (ver Ecuación (6.1)).

$$\boldsymbol{ECM} = \sqrt{\frac{\sum(\boldsymbol{X_i}-\overline{\boldsymbol{X}})}{\boldsymbol{n(n-1)}}} \qquad (6.1)$$

6.2.1 Cálculo del error para la disponibilidad de los servicios

El experimento está diseñado de la siguiente manera: para determinar el comportamiento de los servicios Web en función de su disponibilidad, se tomaron mediciones cada diez ejemplos de aprendizaje. Las mediciones se hicieron para los 10 servicios Web evaluando su comportamiento los días martes y jueves a las 11am y 5pm para los dos días.

Cada uno de los servicios Web, se implementó con un modelo de distribución normal para controlar el comportamiento de su disponibilidad. La tabla de disponibilidad simulada para cada servicio se presenta en la tabla 6-1.

Como se mencionó anteriormente, para esta prueba, se calculó el error cuadrático medio que relaciona el comportamiento de las observaciones para cada servicio y el valor esperado definido en la tabla 6-1.

Tabla 6-1: Valores estimados para los valores de disponibilidad de los servicios en estudio

Servicios	Disponibilidad de los Servicios			
	Martes		Jueves	
	11am	5pm	11am	5pm
BuyItem001	6%	68%	75%	39%
BuyItem002	66%	13%	10%	7%
BuyItem003	36%	78%	64%	76%
BuyItem004	51%	17%	10%	15%
BuyItem005	10%	57%	34%	77%
GetItemInformation001	78%	19%	76%	71%
GetItemInformation002	48%	7%	13%	10%
GetItemInformation003	25%	26%	74%	49%
GetItemInformation004	11%	11%	36%	55%
GetItemInformation005	5%	39%	14%	9%

La valoración para el día martes a las 11am para los servicios evaluados, muestra un comportamiento en el que el sistema de aprendizaje presenta una tendencia a la estabilización a partir de la utilización de 30 ejemplos de entrenamiento como se observa en la figura 6-1 y 6-2. Estas figuras, contiene la evaluación para los 10 servicios implementados para el dominio del shopping.

Cuando se analiza la situación valorando los resultados para el día martes a las 5pm, para los servicios listados en la tabla 6-1 se observa que el modelo de aprendizaje se estabiliza en un rango de 20 a 40 ejemplos de entrenamiento.

Figura 6-1: Comportamiento del error (ECM) del cálculo del riesgo en la métrica de disponibilidad por parte del aprendizaje para el servicio buyItem para el día martes 11am.

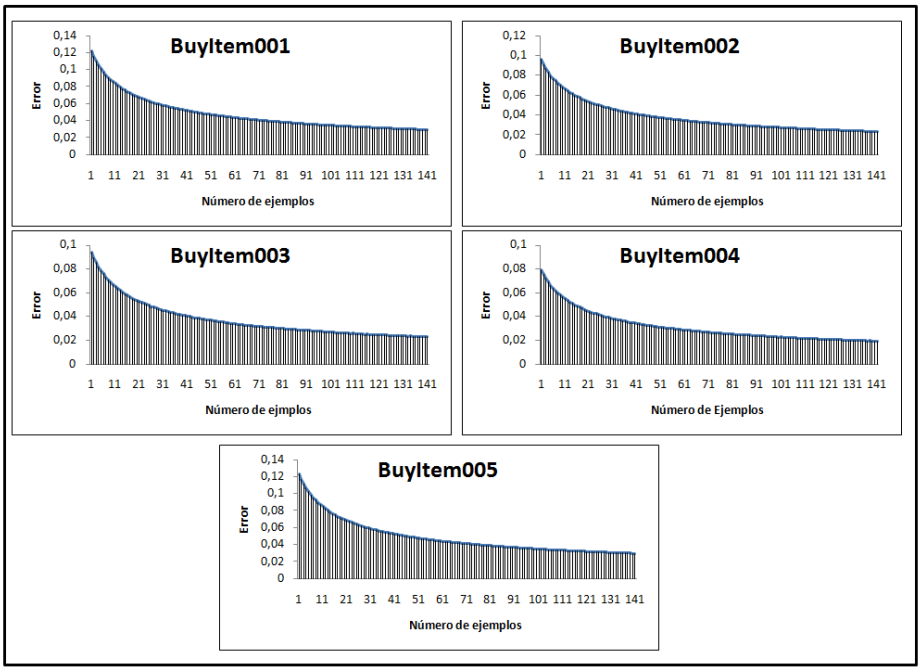
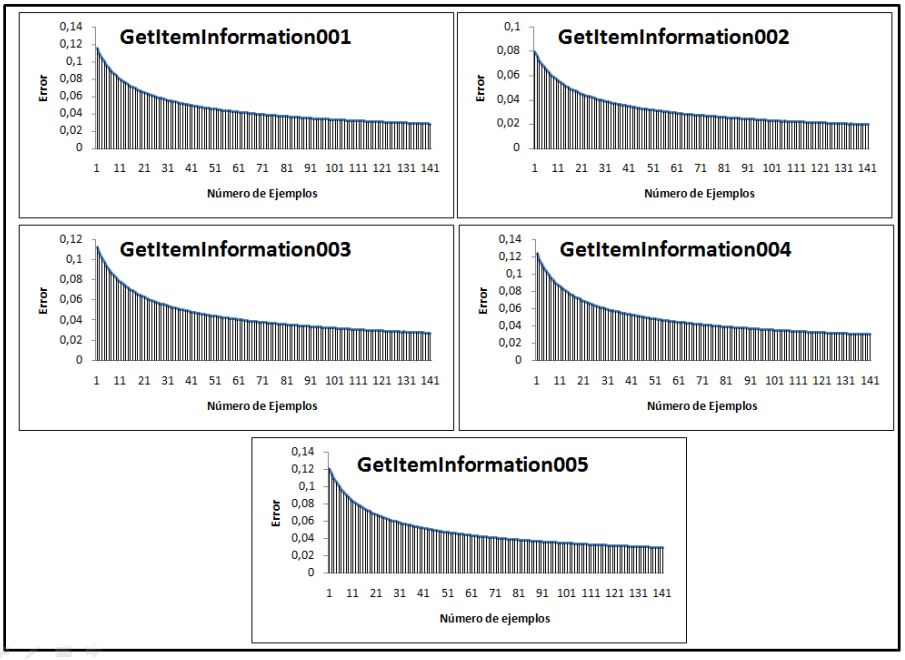


Figura 6-2: Comportamiento del error (ECM) del cálculo del riesgo en la métrica de disponibilidad por parte del aprendizaje para el servicio GetItemInformation para el día martes 11am.



Para el caso del día jueves, los servicios que se evaluaron tanto a las 11am como a las 5pm permitieron verificar que el modelo de aprendizaje se estabiliza cuando se ejecuta con más de 40 ejemplos de entrenamiento.

En la figura 6-2, se observa que el modelo de aprendizaje permite predecir el comportamiento de los servicios utilizados en un plan de composición a partir del entrenamiento con 30 datos de observación de los servicios.

La estabilidad demostrada en el modelo de aprendizaje hace posible la calificación y clasificación por orden de importancia de acuerdo con la disponibilidad de los servicios, y permite la generación de las sugerencias para que el compositor de servicio, utilice aquellos servicios con menor riesgo de falla durante la composición.

La evaluación referente al día martes a las 11am, permite identificar que de los servicios orientados a la compra del artículo, los que se acercan más rápido al valor real a medida que se aumenta el número de ejemplos de entrenamiento, son buyItem002 y buyItem004. Lo que representa que el aprendizaje adquirido sobre el comportamiento de estos dos servicios, es más sólido que el aprendizaje sobre los demás servicios.

En el anexo E, se encuentran las tablas en las que se asocia los valores de error para cada servicio en los días martes 5pm y jueves 11am y 5pm

6.2.2 Cálculo del error para la reactividad de los servicios Web

Al igual que para la disponibilidad, en este experimento se utilizaron los 10 servicios Web descritos anteriormente y la evaluación se realizó los días martes y jueves a las 11am y 5pm, con restricciones de tiempo de 1, 5 y 10 segundos.

Este experimento muestra el error del modelo de aprendizaje al evaluar la reactividad de los servicios. Para esta evaluación se obtuvieron datos de aprendizaje a partir de cada 10 ejemplos de entrenamiento.

La tabla 6-2, presenta los valores simulados por medio de una distribución normal para cada servicio los cuales representan el comportamiento real del servicio.

Tabla 6-2: Valores estimados para los valores de reactividad de los servicios en estudio

Servicios	Reactividad de los Servicios											
	Martes						Jueves					
	11am			5pm			11am			5pm		
	1	5	10	1	5	10	1	5	10	1	5	10
BuyItem001	57 %	64 %	84 %	8%	57%	62%	10%	33%	64%	8%	39%	54%
BuyItem002	13 %	58 %	84 %	8%	42%	47%	35%	47%	80%	15%	37%	40%
BuyItem003	57 %	63 %	81 %	36%	43%	83%	48%	48%	78%	44%	63%	72%
BuyItem004	46 %	56 %	73 %	6%	40%	50%	8%	56%	67%	20%	34%	35%
BuyItem005	9%	14 %	48 %	5%	56%	80%	21%	59%	77%	41%	54%	67%
GetItemInformation001	58 %	60 %	78 %	23%	38%	56%	40%	60%	66%	19%	32%	43%
GetItemInformation002	16 %	31 %	82 %	17%	34%	40%	38%	50%	63%	30%	47%	56%
GetItemInformation00	51	55	77	44%	48%	72%	49%	51%	51%	13%	21%	34%

Servicios	Reactividad de los Servicios											
	Martes						Jueves					
	11am			5pm			11am			5pm		
	1	5	10	1	5	10	1	5	10	1	5	10
3	%	%	%									
GetItemInformation004	15 %	21 %	60 %	15%	40%	55%	17%	45%	59%	48%	50%	52%
GetItemInformation005	7%	20 %	46 %	28%	38%	63%	51%	65%	69%	29%	34%	40%

En las figuras 6-3 y 6-4, se observa el valor del error cuadrático medio calculado para la métrica de reactividad de los 10 servicios mencionados.

Figura 6-3: Comportamiento del error (ECM) del cálculo del riesgo en la métrica de disponibilidad por parte del aprendizaje para el servicio BuyItem para el día martes 11am.

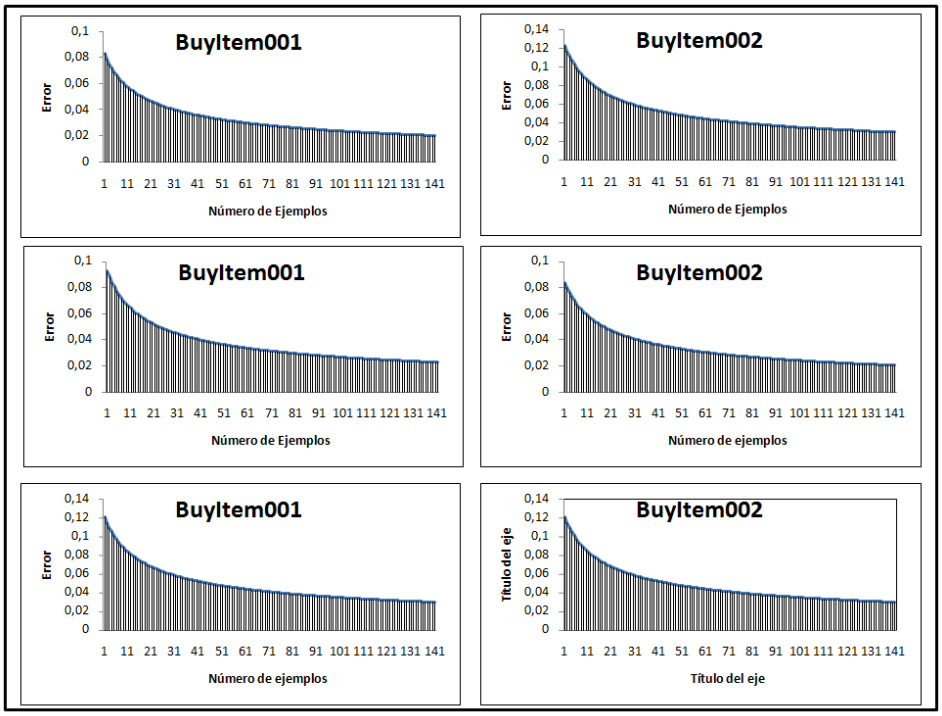
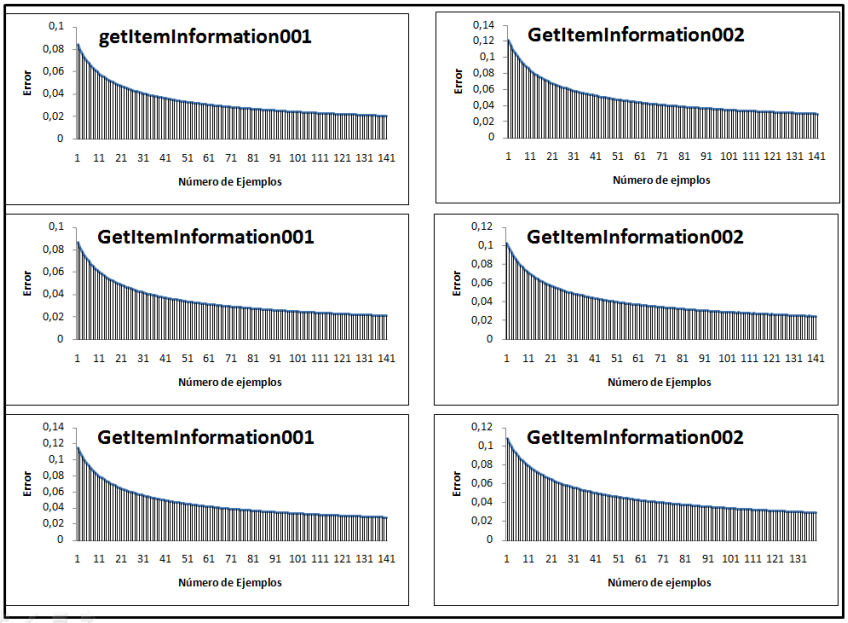


Figura 6-4: Comportamiento del error (ECM) del cálculo del riesgo en la métrica de disponibilidad por parte del aprendizaje para el servicio GetItemInformation para el día martes 11am.



La valoración para el día martes a las 11am para los servicios evaluados, indica que el sistema de aprendizaje se estabiliza a partir de la utilización de 40 ejemplos de entrenamiento como se observa en la figura 6-3.

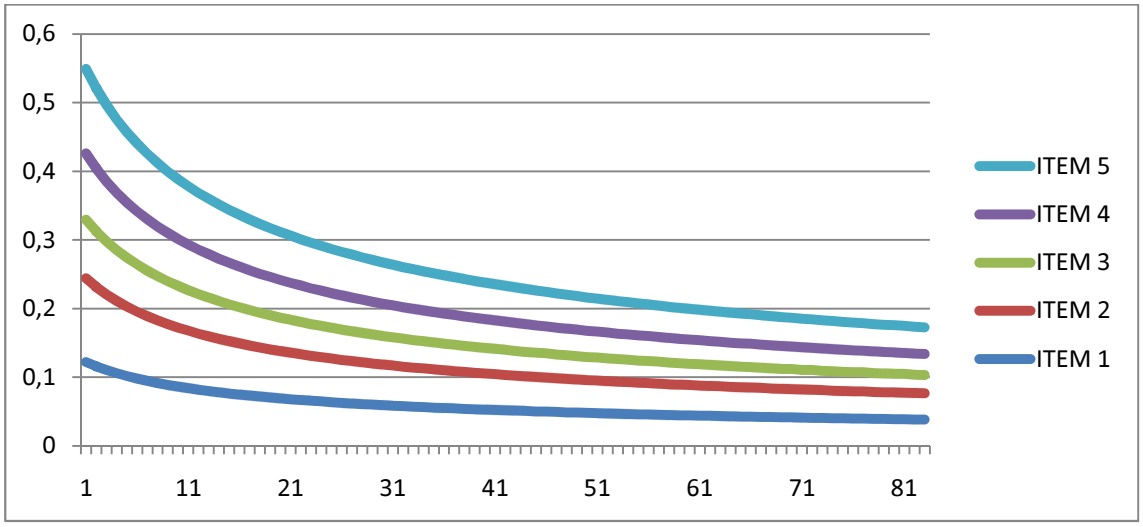
Para el caso de las evaluaciones realizadas el día jueves, la estabilización del sistema de aprendizaje se presentó en el rango de 30 a 40 ejemplos de entrenamiento.

Estos resultados demuestran que el comportamiento de los servicios de acuerdo a su reactividad son aprendidos satisfactoriamente por el modelo de aprendizaje el cual a partir de cada ejecución, obtiene el conocimiento de control que permitirá guiar al compositor en la selección del conjunto de los mejores servicios representados a través de su reactividad y tratando de disminuir los fallos en la composición.

6.2.3 Cálculo del error para manejo de creencias

Para la evaluación del error de aprendizaje para este criterio, se utilizó el servicio Web getItemInformation001 y para eso, se generaron 5 ítems que representan las creencias del compositor. La medición consistió en evaluar el comportamiento que tienen estas creencias y determinar su error. La figura 6-5 presenta el comportamiento del error para los cinco ítems.

Figura 6-5: Comportamiento del error (ECM) del cálculo del riesgo en la métrica manejo de creencias por parte del aprendizaje para el servicio GetItemInformation



Como se observa en la figura 6-5 existe una convergencia en el conocimiento de las creencias a medida que estas se observan en mayor número. Estas permiten identificar una estabilidad demostrada en el sistema de aprendizaje que hace posible la generación de las sugerencias para el compositor de servicio aumente su factor de certeza.

La evaluación vista en la figura 6-5 representa el aprendizaje adquirido sobre el manejo de las creencias y se concluye entonces la solides del aprendizaje y su capacidad para la interpretación del manejo de las mismas.

Así el modelo sugerido demuestra la capacidad para encontrar asociaciones correctas de acuerdo con las creencias del compositor y el incremento de esa capacidad mientras más observaciones se presenten, mostrando una relativa estabilidad a partir de 30 observaciones.

6.3 Evaluación del riesgo

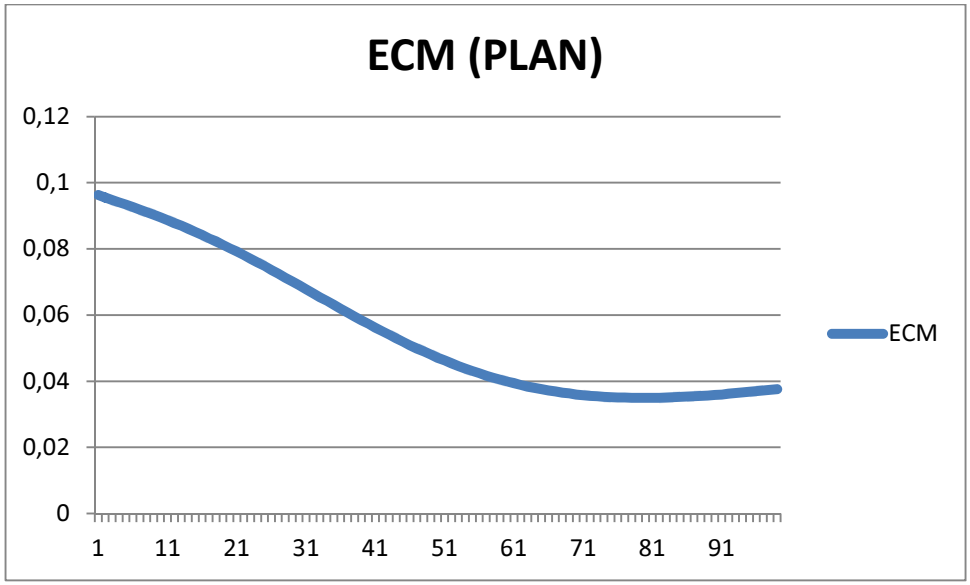
Una vez definidos los valores de disponibilidad y reactividad de los servicios estudiados para el caso de compras por internet y demostrada la eficiencia y certeza del sistema de aprendizaje se procedió a realizar las simulaciones correspondientes para la selección de los mejores servicios para la composición del plan, calcular el riesgo conjunto o riesgo del plan y verificarlo mediante el cálculo del error cuadrático medio

Usando la ecuación 2.10 propuesta en el capítulo 2 de esta tesis, se calculó el riesgo para cada servicio en estudio al multiplicar sus valores de disponibilidad y reactividad aprendidos, por los pesos asignados según la ecuación 2.8.

Una vez definidos los riesgos se seleccionan los servicios que presenten los menores valores y que por tanto disminuyen las fallas en la composición. Se calculó el riesgo conjunto de los servicios para diferente número de observaciones y se calculó el error cuadrático medio de la estimación de riesgo realizada.

A continuación, la figura 6-6 se presenta la gráfica que representa el cálculo del error cuadrático medio para el caso estudiado anteriormente (10 servicios, Martes 11am)

Figura 6-6: Calculo del Error del Plan.



Como se observa en la figura de riesgo del plan, cuando el número de ejemplos de entrenamiento es bajo, el desconocimiento del sistema de aprendizaje se refleja en valores bajos y errados del riesgo, lo cual se verifica al observar la tabla del error del riesgo del plan calculado Tabla 6-3. Sin embargo, a medida que el número de ejemplos de entrenamiento aumenta el valor del riesgo tiende a estabilizarse y se comprueba, desde la tarea del compositor el correcto funcionamiento del modelo de aprendizaje del sistema.

Tabla 6-3: Valor del riesgo del plan para distinto número de observaciones

Numero de observaciones	Riesgo del plan
1	0.0688
10	0.9778
20	2.7525
30	5.3836
40	8.5698
50	11.7204
60	14.263
70	15.9361
80	16.8332
90	17.2247
100	17.3636

Como se observa en la tabla 6-3, el valor del riesgo a medida que el sistema aprende se va estabilizando, lo cual es claro al analizar las últimas filas de la tabla 7.3 donde las diferencias entre los datos va disminuyendo. Ese comportamiento, demuestra que a medida que el modelo tiene mayor información sobre los servicios realiza un mejor aprendizaje y genera mejores reglas alrededor del comportamiento de los servicios.

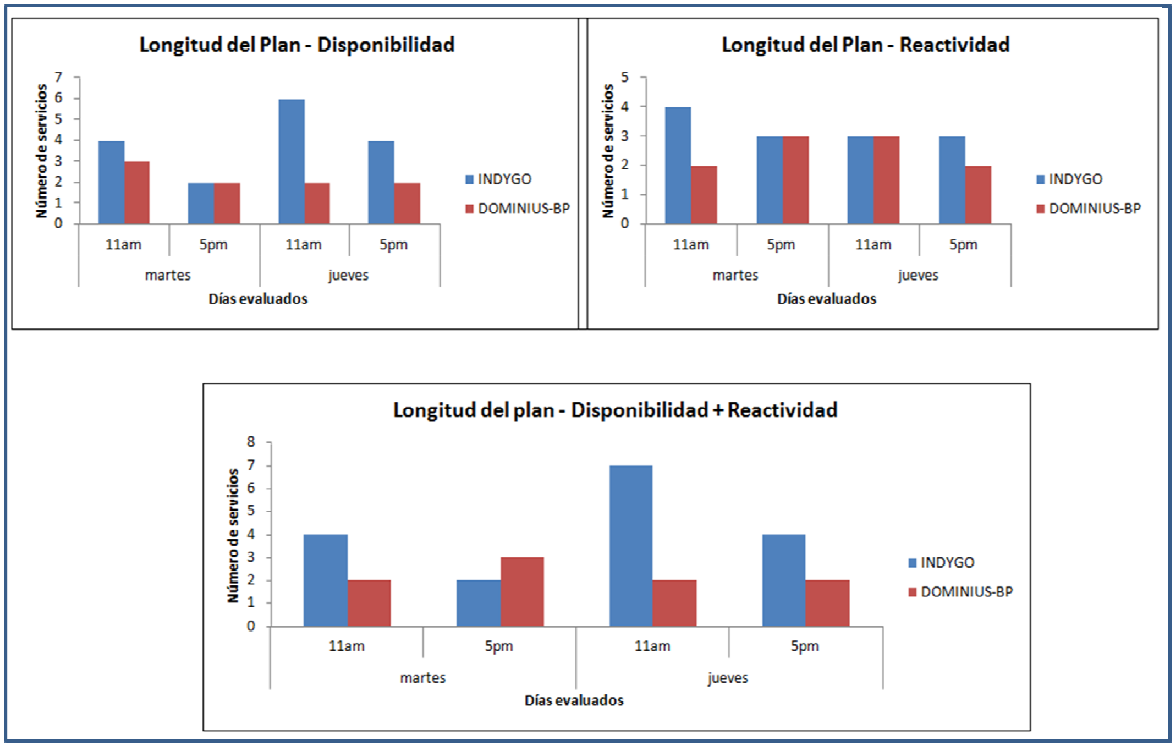
6.4 Longitud del Plan de Composición

La ventaja de contar con el conocimiento de control, es omitir algunos pasos innecesarios para alcanzar los objetivos. En otras palabras, seleccionar aquellos servicios que tienen menor probabilidad de riesgo y sugerirlos al compositor de servicios para realizar la composición.

Este experimento, consistió en evaluar el comportamiento del sistema de composición de servicios cuando este, hace uso del conocimiento de control emitido por el modelo de aprendizaje y sin dicho conocimiento. Para ello, se utilizaron los 10 servicios Web descritos anteriormente y se evaluó el alcance del plan de composición en diferentes días y horas de ejecución.

Los días evaluados fueron martes y jueves a las 11am y 5pm. El comportamiento del sistema de composición cuando se hace uso de la disponibilidad y la reactividad del servicio se describe en la figura 6-7

Figura 6-7: Longitud del plan.



Esta figura, muestra que el compositor de servicios recurre al uso de servicios fallidos para conseguir el plan de composición. El conocimiento de control, le permite al compositor utilizar aquellos servicios que mejor se comporten en un determinado instante en el tiempo. Es así como en la figura se observa la variación de servicios que el compositor utiliza para alcanzar un plan. Sin embargo, haciendo uso de la información del comportamiento de los servicios como lo es su disponibilidad y reactividad, el compositor disminuye el conjunto de servicios a utilizar para alcanzar el plan.

6.5 Discusión

Después de realizar el conjunto de pruebas para validar el modelo de aprendizaje, se pudo identificar que el uso de criterios de falla apoyado en el aprendizaje de máquinas, aporta un gran valor para disminuir las fallas ocasionadas en la composición de servicios ya que la adquisición automática de la información, permite acercarse más al comportamiento real de los servicios.

El modelo de aprendizaje tiene una tendencia convergente a medida que aumenta el número de ejemplos de entrenamiento, de tal manera que para los servicios en estudio, a medida que se aumentan esos ejemplos de entrenamiento disminuye el nivel de error de los supuestos que aprende el sistema y que permitirán proceder con la selección de los servicios a usar por el compositor.

Por otro lado, en el error del riesgo calculado para el plan se observa como al incrementar el número de ejemplos de entrenamiento el error disminuye, pues mejora el aprendizaje y los valores estimados por el sistema se aproximan cada vez más a los valores reales del plan.

Este conjunto de valoraciones, hizo posible comprobar la susceptibilidad en la que están envueltos los servicios Web, y como su comportamiento afecta directamente la composición de servicio.

7. Conclusiones y recomendaciones

7.1 Conclusiones

Esta tesis, propone un modelo de aprendizaje de máquinas para adquirir automáticamente el conocimiento relacionado con el factor de riesgo de los servicios, evaluando la participación de cada servicio como un todo dentro del plan final de composición, tratando de minimizar el riego de falla de la composición.

Para alcanzar el desarrollo completo de este modelo, se hicieron algunos trabajos específicos que dieron origen a los siguientes aportes.

En primer lugar, se desarrolló el sistema DOMINIUS_BP, el cual implementa y valida el modelo de aprendizaje propuesto. Este sistema, permite la manipulación de tres criterios que identifican los riesgos de falla asociados a una composición de servicios. Estos criterios son: la disponibilidad, la reactividad y el manejo de creencias; los dos primeros, hacen referencia a la evaluación del riesgo de falla calculado en el entorno de los servicios mientras que el último criterio, se orienta a determinar el riesgo de falla ocasionado por el compositor de servicios.

En segundo lugar, por cada criterio que identifica el riesgo de falla, se desarrollaron dos traductores. El primero, captura la información de las ejecuciones de los servicios y extrae de ésta, la información relevante para cada criterio y lo transforma en un lenguaje de aprendizaje de máquinas (lenguaje lógico), en la que se identifican dos partes fundamentales para el aprendizaje que son: la base de conocimientos y la función objetivo. A partir de estos datos, se aplica los algoritmos de aprendizaje para obtener las reglas de aprendizaje que representan el conocimiento de control para cada servicio ejecutado obteniendo su riesgo de falla en función del plan de composición. Cuando estos datos se generan, el segundo traductor es el encargado de interpretar las reglas de aprendizaje producidas para cada criterio y las traslada a una base de datos, la cual almacena el conocimiento de control para futuras composiciones.

En tercer lugar, se diseñó e implementó una ontología que permite describir el factor de riesgo ocasionado en el entorno del servicio. Esta ontología, se realizó bajo las especificaciones de la ontología OWL-S en la cual, se extendió el perfil del servicio (serviceprofile) a través de la clase (serviceparameter) a la que se le adicionó un conjunto de parámetros que representan los criterios de disponibilidad y reactividad de los servicios. La ontología para describir el riesgo, asocia semánticamente el valor de cada criterio y el impacto que éste tiene en el servicio. Por lo tanto, cuando se requiere identificar el riesgo de falla de un servicio Web, se crea una ontología extendida del serviceprofile en la cual se especifique los parámetros que representan los criterios de falla que se quieren asociar al servicio.

En cuarto lugar, se implementó el componente de integración entre el sistema de aprendizaje DOMINIUS-BP y una técnica de planificación orientada a la composición de servicios INDYGO. Para esta integración, este componente abstrae la lista de servicios que se requieren para hacer un plan de composición y por cada uno construye una nueva representación ontológica en la que se especifica el valor de riesgo dependiendo del criterio que se evalúe y el impacto asociado al criterio. El valor del riesgo se obtiene a través de las reglas almacenadas en la base de datos producto del sistema de aprendizaje.

En quinto lugar se diseñó y desarrolló un ambiente de pruebas para validar el modelo propuesto. Este ambiente, tuvo como dominio de composición al dominio de compras por internet. Para este dominio, se implementaron 50 servicios Web atómicos reales y a los cuales, se les generó una descripción semántica utilizando la ontología owl.

El sexto lugar, representa la validación del modelo propuesto a través de la recopilación de las evaluaciones, producto de la valoración que se hizo a cada servicio en función de los criterios de riesgo y a través de los cuales, se identificó el aporte del aprendizaje de máquinas para la captura de la información del riesgo de falla para estos diferentes criterios y como este aprendizaje ayuda en la selección de servicios para disminuir el riesgo de falla en las composiciones de servicios.

7.2 Recomendaciones

Dentro del marco de esta tesis, surgen aspectos que se apoyan en este trabajo para ser parte de trabajos futuros.

Teniendo en cuenta la valoración de los riesgos, uno de los trabajos futuros es la identificación de patrones de riesgos ocasionados por las políticas de seguridad de los servicios. Éste, es de gran información para el manejo de fallas puesto que este criterio proporciona una semántica característica de los posibles problemas que no soporta un servicio y que son ajenos al conocimiento del compositor; de manera que si se identifica los valores producidos por este criterio, determinaría con más precisión, cuando utilizar el servicio en el plan, mejorando la composición.

Igualmente, se propende por utilizar criterios que midan el riesgo de falla, utilizando valores adicionales que no sean parte del criterio pero que sí lo afecte, para mejorar su expresividad. En el caso de este trabajo, se utilizó la temporalidad como valor adicional de cálculo; sin embargo, es importante hacer uso de medidas como impactos semánticos o relaciones de satisfacción de objetivos de composición. De esta manera, los valores obtenidos por los criterios tienden a ser más objetivos.

El manejo de fallas, se resume en la utilización de los componentes principales de un servicio, es decir omitir información innecesaria y generar nuevas representaciones de servicios a través de un conjunto de reglas de aprendizaje, que permitan identificar las relaciones más importantes de los servicios descritos en sus precondiciones y efectos y a partir de allí, guiar al compositor en la selección de los mejores servicios para integrarlos en el plan de composición.

Finalmente, aunque en este trabajo de tesis se implementó una ontología para el manejo de fallas, es importante que ésta sea ampliada con nuevos criterios relacionados con los servicios y con los compositores; así se estandariza una ontología de fallas capaz de cubrir y representar el marco total de los servicios y mejorar la interpretación semántica de éstos.

A. Anexo: Selección del criterio y cálculo de su impacto asociado

Este anexo, presenta las pruebas realizadas para obtener el criterio de mayor importancia en el cálculo de riesgo de falla en la composición de servicios. Además, presenta las pruebas para la selección de la mejor técnica de asignación de impactos para el cálculo de impactos a los criterios.

A.1 Selección del criterio con mayor importancia

Para identificar cual es el criterio de mayor importancia en la composición de servicios referentes al riesgo, se utilizó la heurística de entropía y se realizó la siguiente prueba:

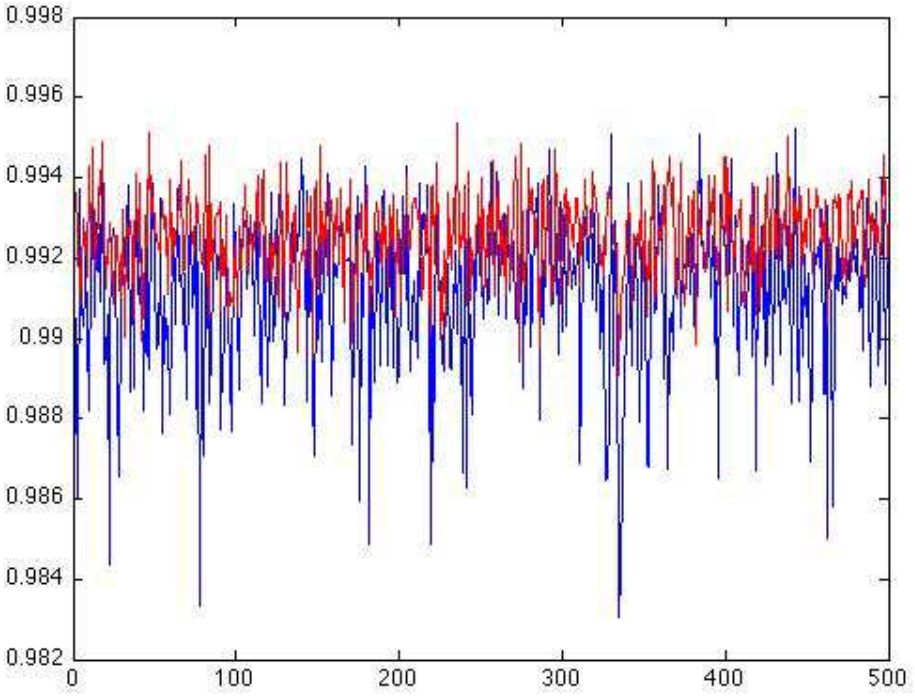
Para cada variable de riesgo (disponibilidad y reactividad) se asignaron 10000 valores que representan el riesgo de falla para cada criterio. Dichos valores se estimaron en un rango porcentual [0-100]. Adicional a esto, se obtuvo el valor de ejecutar los valores de estos criterios en un servicio es decir, si su ejecución fue éxito o falla.

La prueba consistió en encontrar el criterio que mejor represente la entropía; es decir, el criterio que en sus cálculos disminuya el valor de la entropía. Para esto se realizaron 500 iteraciones cada una evaluando los 10000 registros de cada criterio.

Una vez ejecutada la prueba, se encuentra que en 367 casos que en términos porcentuales representan el 73.4% de los ensayos, la disponibilidad presentó un valor de entropía menor al de la reactividad lo cual demuestra un mayor orden en los datos y lo convierte en el atributo que mejor separa (ordena) la información.

En la figura A-1, se observa el resultado para el cálculo de la entropía en las 500 iteraciones realizadas. La línea roja representa la reactividad y la línea azul la disponibilidad. Es clara la tendencia de la disponibilidad a presentar un valor inferior de entropía que la reactividad. Por lo tanto, el criterio de mayor importancia en el cálculo de riesgo es la **disponibilidad**.

Figura A-1: Cálculo de la entropía para los criterios de disponibilidad y reactividad



A.2 Selección de la técnica de asignación de impactos a los criterios

Una vez identificado el criterio con más importancia en el cálculo del riesgo, se procede a calcular el valor del impacto asociado a dicho criterio. Para ello, los trabajos estudiados en el estado del arte, referencian técnicas como ROC (Rank-order centroid), RR (Rank reciprocal) y RS (Rank sum) para realizar dicho cálculo.

Con el fin de definir cuál es la técnica que mejor describe la minimización del riesgo, se calcularon los valores de los impactos para cada criterio a partir de cada técnica y se identifica la minimización del riesgo a través de la ecuación 2.6.

La prueba consistió en hacer 100 ejecuciones con 10000 registros y calcular a partir de los impactos asociados por cada técnica a cada criterio, la minimización del riesgo total de ejecución. En la tabla A-1 se relacionan los resultados obtenidos para cada técnica al evaluar los criterios de disponibilidad y reactividad de los servicios.

Las técnicas RR y RS al utilizar dos criterios, asigna el mismo impacto para cada criterio por lo tanto se optó por tomar la técnica RR para hacer la comparación entre técnicas de adquisición del valor de impacto.

Como resultado, se obtiene que la técnica que mejor asigna los impactos a los criterios disponibilidad y reactividad para minimizar el riesgo de falla es: RR asignando los impactos (0.67) a la disponibilidad y (0.33) a la reactividad de los servicios.

Tabla A-1: Comparación de técnicas de asignación de impactos

TÉCNICA	IMPACTOS		Minimización del Riesgo Total
	Disponibilidad	Reactividad	
ROC	0,75	0,25	45
RR	0,67	0,33	55

B. Anexo: Ontología QoS para representación de fallas

Este anexo, describe la ontología utilizada por el modelo de aprendizaje para marcar semánticamente el riesgo que presenta un servicio en la composición, en función de dos criterios disponibilidad y reactividad.

B.1 Representación ontológica OWL

B.1.1 Clases

La ontología QoS está constituida por tres clases: una clase principal denominada QoS, la cual contiene una clase denominada Non_Functional_Parameter y ésta a su vez contiene las clases que representan los criterios de riesgos de falla para cada servicio es decir Availability y Reactivity.

- ```
  <owl:Class rdf:ID="QoS"/>
  ```
- ```
  <owl:Class rdf:ID="Non_Functional_Parameter"/>
  ```
- ```
  <owl:Class rdf:ID="Availability">
    <rdfs:subClassOf>
  <owl:Class rdf:about="Non_Functional_Parameter"/>
  </rdfs:subClassOf>
  </owl:Class>
  ```
- ```
  <owl:Class rdf:ID="Reactivity">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Non_Functional_Parameter"/>
    </rdfs:subClassOf>
  </owl:Class>
  ```

B.1.2 Propiedades

Esta ontología contiene dos propiedades de tipo DataTypeProperty: la primera non_funtional_parameterHasValue, asocia un valor numérico a cada criterio el cual representa el riesgo del servicio; y la segunda non_funtional_parameterHasWeight, asigna el valor numérico del impacto asociado a dicho criterio.

- ```
  <owl:onProperty>
         <owl:DatatypeProperty rdf:ID="non_funtional_parameterHasValue"/>
  </owl:onProperty>
  ```
- ```
  <owl:onProperty>
         <owl:DatatypeProperty rdf:about="non_funtional_parameterHasWeight"/>
  ```

```
</owl:onProperty>
```

B.1.3 Restricciones

Las restricciones que se definen en la ontología QoS son dos: cardinalidad mínima y cardinalidad máxima. Estas restricciones están asociadas a las propiedades descritas anteriormente las cuales, obligan a dichas propiedades a tener una instancia como mínimo y máximo.

- ```
  <owl:Restriction>
     <owl:onProperty>
        <owl:DatatypeProperty rdf:about="non_funtional_parameterHasValue"/>
      </owl:onProperty>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:m
      axCardinality>
  </owl:Restriction>
  ```

- ```
  <owl:Restriction>
     <owl:onProperty>
        <owl:DatatypeProperty rdf:about="non_funtional_parameterHasValue"/>
     </owl:onProperty>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:m
      axCardinality>
  </owl:Restriction>
  ```

- ```
  <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:m
      inCardinality>
     <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="non_funtional_parameterHasWeight"/>
     </owl:onProperty>
  </owl:Restriction>
  ```

- ```
  <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about=" non_funtional_parameterHasWeight"/>
      </owl:onProperty>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:m
      axCardinality>
  </owl:Restriction>
  ```

C. Anexo: Ejemplos de la ontología QoS para un servicio

En este anexo, se presenta un conjunto de ejemplos del perfil del servicio buyItem_008ms, asociándole los datos de riesgo.

C.1 Ejemplos de la ontología profile incorporando valores de riesgo

La figura C-1, presenta el perfil del servicio buyItem_008ms con disponibilidad como parámetro de riesgo, aquí se observa que el valor del riesgo de disponibilidad es de 80% y su impacto es 1

Figura C-1: Perfil del servicio buyItem_008ms con riesgo de disponibilidad

```
▼<service:presents>
 ▼<profile:Profile rdf:about="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#buyItem_008msProfile">
    <profile:hasInput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#itemdata"/>
   ▼<profile:serviceParameter>
    ▼<rdf:Description rdf:about="Availability_1">
       <qos:non_funtional_parameterHasValue>0,8</qos:non_funtional_parameterHasValue>
       <qos:non_funtional_parameterHasWeight>1</qos:non_funtional_parameterHasWeight>
     </rdf:Description>
    </profile:serviceParameter>
    <profile:textDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Buy an item from the vendor store</profile:textDescription>
    <profile:hasInput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#ccexp"/>
    <profile:hasPrecondition rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#XSPDDL-Precondition"/>
    <profile:has_process rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#buyItem_008msAtomicProcess"/>
    <profile:hasInput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#cc"/>
    <profile:hasOutput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#result"/>
    <service:presentedBy rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#buyItem_008msService"/>
    <profile:serviceName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">buyItem_008ms</profile:serviceName>
  </profile:Profile>
</service:presents>
```


La figura C-2 presenta el perfil del servicio buyItem_008ms con reactividad como parámetro de riesgo. Esta figura, muestra que la reactividad del servicio tiene un riesgo de 60% y su impacto es 1

Figura C-2: Perfil del servicio buyItem_008ms con riesgo de reactividad

```
▼<service:presents>
 ▼<profile:Profile rdf:about="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#buyItem_008msProfile">
    <profile:hasInput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#itemdata"/>
    <profile:textDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Buy an item from the vendor store</profile:textDescription>
    <profile:hasInput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#ccexp"/>
    <profile:hasPrecondition rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#XSPDDL-Precondition"/>
    <profile:has_process rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#buyItem_008msAtomicProcess"/>
    <profile:hasInput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#cc"/>
   ▼<profile:serviceParameter>
    ▼<rdf:Description rdf:about="Reactivity_1">
       <qos:non_funtional_parameterHasValue>0,6</qos:non_funtional_parameterHasValue>
       <qos:non_funtional_parameterHasWeight>1</qos:non_funtional_parameterHasWeight>
     </rdf:Description>
    </profile:serviceParameter>
    <profile:hasOutput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#result"/>
    <service:presentedBy rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#buyItem_008msService"/>
    <profile:serviceName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">buyItem_008ms</profile:serviceName>
  </profile:Profile>
</service:presents>
```

La figura c-3 presenta el perfil del servicio buyItem_008ms con dos parámetros de riesgo. Aquí, la disponibilidad del servicio tiene un riesgo asociado del 80% y un impacto de 0.67 mientras que la reactividad del servicio tiene un riesgo de falla de 60% con un impacto de 0.33.

Figura C-3: Perfil del servicio buyItem_008ms con riesgo de disponibilidad y reactividad

```
▼<service:presents>
 ▼<profile:Profile rdf:about="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#buyItem_008msProfile">
    <profile:hasInput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#itemdata"/>
   ▼<profile:serviceParameter>
    ▼<rdf:Description rdf:about="Availability_1">
       <qos:non_funtional_parameterHasValue>0,8</qos:non_funtional_parameterHasValue>
       <qos:non_funtional_parameterHasWeight>0,67</qos:non_funtional_parameterHasWeight>
     </rdf:Description>
    </profile:serviceParameter>
    <profile:textDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Buy an item from the vendor store</profile:textDescription>
    <profile:hasInput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#ccexp"/>
    <profile:hasPrecondition rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#XSPDDL-Precondition"/>
    <profile:has_process rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#buyItem_008msAtomicProcess"/>
    <profile:hasInput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#cc"/>
   ▼<profile:serviceParameter>
    ▼<rdf:Description rdf:about="Reactivity_1">
       <qos:non_funtional_parameterHasValue>0,6</qos:non_funtional_parameterHasValue>
       <qos:non_funtional_parameterHasWeight>0,33</qos:non_funtional_parameterHasWeight>
     </rdf:Description>
    </profile:serviceParameter>
    <profile:hasOutput rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#result"/>
    <service:presentedBy rdf:resource="http://localhost/indygo/Domains/ShoppingDomain008/Services/buyItem_008ms.owl#buyItem_008msService"/>
    <profile:serviceName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">buyItem_008ms</profile:serviceName>
  </profile:Profile>
 </service:presents>
```

D. Anexo: Manual del sistema DOMINIUS-BP

Este anexo, presenta el manual de usuario del sistema DOMINIUS-BP, detallando paso a paso cada componente y su funcionamiento.

D.1 Descripción general

El software desarrollado en este trabajo de tesis, tiene como función principal adquirir la información del factor de riesgo de los servicios de forma automática. Para esto, implementa un conjunto de componentes que ayudan a describir y entender el proceso de aprendizaje. Entre las funciones principales de este sistema están:

Generación automática de los ejemplos de entrenamiento (observaciones de la ejecución de los servicios).
Generación de los archivos de aprendizaje
Integra un sistema de composición de servicios con el sistema de aprendizaje, utilizando una nueva representación de los servicios es decir, adicionándoles el conocimiento de control.

Esta información, se encuentra localizada en la ventana inicial del software, donde se le pregunta visualmente al usuario que es lo que quiere hacer. La figura D-1, muestra la ventana principal del sistema DOMINIUS-BP.

Adicionalmente esta ventana contiene un menú de idiomas en el cual se selecciona el lenguaje del sistema entre tres posibles idiomas (español-ingles).

Dependiendo de la utilidad que seleccione el usuario, se desplegarán nuevas ventanas con diferentes opciones para cumplir los requerimientos del usuario. Estas ventanas y su comportamiento se describen a continuación.

Figura D-1: Interfaz gráfica principal del sistema DOMINIUS-BP

D.2 Generación de las observaciones de ejecución

D.2.1 Datos de entrada

El sistema DOMINIUS-BP requiere las siguientes entradas:

Ontología Inicial: es la dirección en la cual se encuentra localizada la ontología inicial. Esta ontología describe el estado inicial del cual partirá el proceso de composición.

Ontología Final: hace referencia a la dirección donde se encuentra localizada la ontología final. Esta ontología representa el estado final que busca el proceso de composición.

Lista de Servicios: es la dirección donde se encuentra ubicada la lista de servicios que participan en el proceso de composición. Esta lista, es un archivo plano donde cada línea contiene la URL donde se encuentra localizado cada servicio.

Restricción de tiempo de planificación: es el tiempo máximo que dispone el compositor para calcular cada una de las acciones del proceso de composición.

Restricción de tiempo de ejecución: es el tiempo máximo que tiene un servicio para dar una respuesta al cliente. Este dato solo se activa cuando el criterio de reactividad está activado.

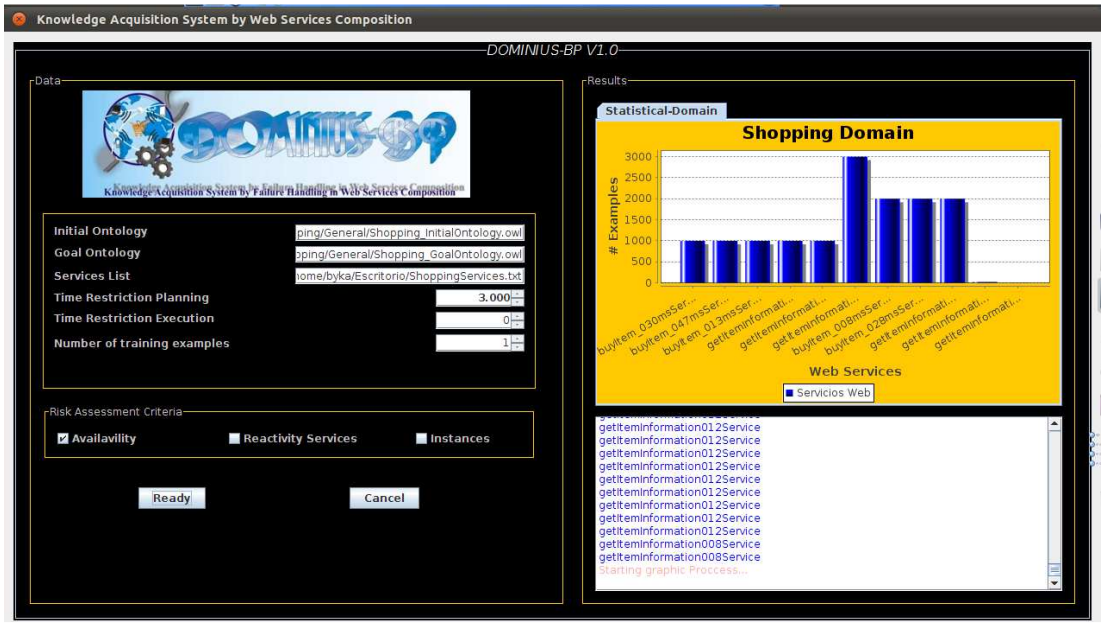
Número de ejemplos de entrenamiento: representa el número de ejemplos de entrenamiento que se desean crear.

Criterios de Riesgo: hacen referencia a la información del servicio que se desea sea capturar en cada uno de los ejemplos de entrenamiento. Estos criterios son: disponibilidad, reactividad y manejo de creencias (availability, reactivity, instances) y se selecciona uno o varios de ellos para capturar la información.

D.2.2 Proceso

Ingresadas las entradas de datos del sistema, existe un botón ready que dará inicio al proceso de generación de los ejemplos de entrenamiento. En esta ventana, existen dos componentes visuales que advierten al usuario sobre lo que está haciendo el sistema. En primer lugar, está un componente de graficación el cual presenta el conjunto de servicios que se ejecutaron y sobre los cuales se genera un ejemplo de aprendizaje. El otro componente es un lienzo que le muestra al usuario el total de los ejemplos que se crearon.

Figura D-2: Interfaz gráfica para generar ejemplos de entrenamiento



Para salir de esta ventana, existe un segundo botón cancel, que cierra la ventana actual y activa la ventana principal del sistema DOMINIUS-BP

D.3. Generación de los archivos de aprendizaje

D.3.1 Datos de entrada

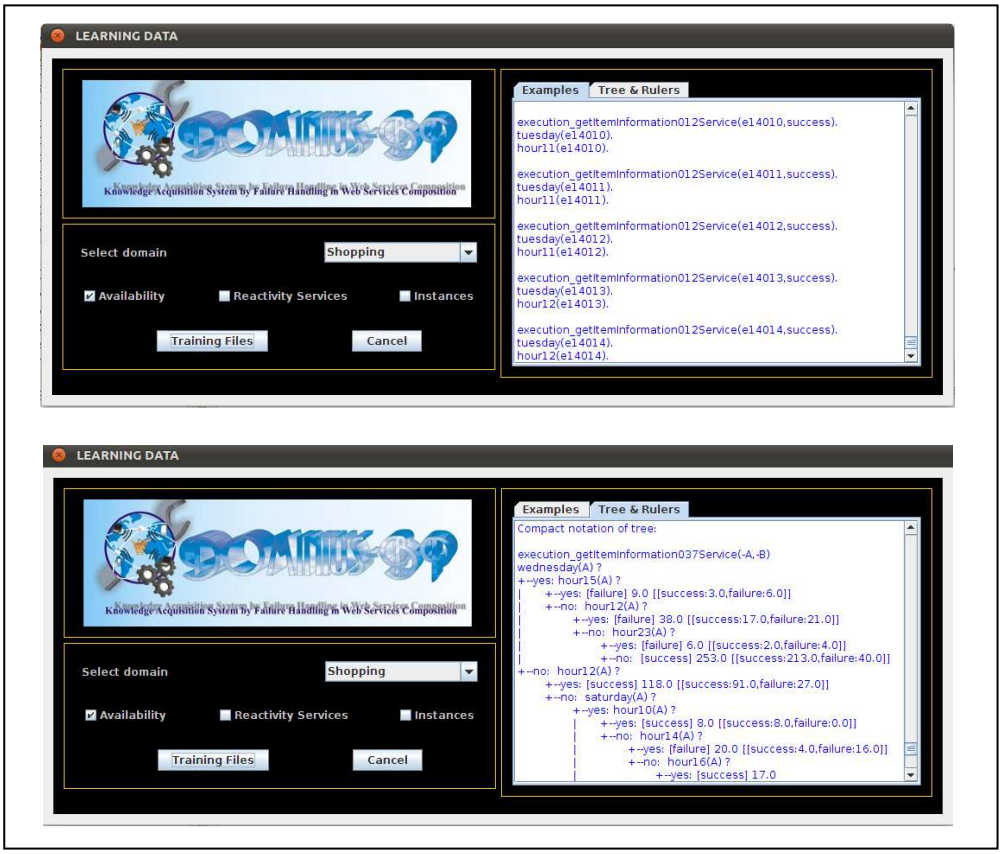
Los datos que son requeridos en esta ventana son:

Dominio: es el nombre del dominio al que se asocia los ejemplos de entrenamiento.
Criterios de Riesgo: hace referencia a los criterios sobre los que se desea aprender. Si existen ejemplos entrenados para el criterio o criterios seleccionados, se habilitará el check del criterio para seleccionarlo; de lo contrario, el check queda inhabilitado.

D.3.2 Proceso

Cuando los datos de entrada son establecidos, existe un botón TrainingFiles que da inicio al proceso de aprendizaje. Como información al usuario, la ventana cuenta con un lienzo el cual, se divide en dos pestañas, la primera contiene los ejemplos de entrenamiento codificados en el lenguaje lógico y la segunda pestaña presenta los árboles de aprendizaje y sus reglas, hecho que se repite por cada servicio evaluado por el aprendizaje.

Figura D-3: Interfaz gráfica para generar archivos de aprendizaje



Para cerrar esta ventana, está el botón cancel que retorna a la ventana principal del sistema DOMINIUS-BP.

D.4 Integra un sistema de composición de servicios con el sistema de aprendizaje

D.4.1 Datos de entrada

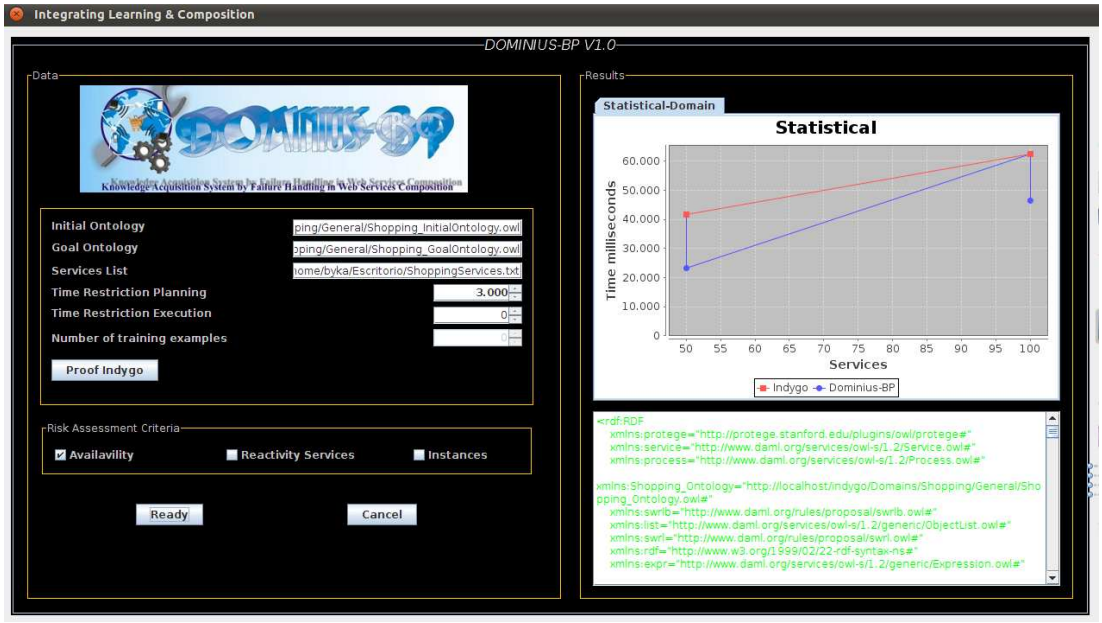
Los datos de entrada para esta ventana son los descritos en D.2.1 omitiendo el número de ejemplos de entrenamiento.

D.4.2 Proceso

En esta ventana, aparece un nuevo botón Indygo el cual ejecuta remotamente el sistema de composición INDYGO con los datos de entrada. Este proceso se utiliza para hacer pruebas comparativas entre el sistema de composición como sistema unitario, con el sistema de composición apoyado por el sistema de aprendizaje. Las pruebas comparativas se obtienen por cada uno de los criterios seleccionados. Los resultados son expuestos en el panel de gráficas.

Por otra parte, el botón ready, da inicio al proceso de integración del sistema de composición con el sistema de aprendizaje. En este caso, el usuario observa en el lienzo, las nuevas representaciones ontológicas de los servicios en las cuales se identifica los valores de riesgo asociados al servicio por cada uno de los criterios seleccionados (disponibilidad y/o reactividad).

Figura D-4: Interfaz gráfica para integrar INDYGO y DOMINIUS-BP



Para salir de esta ventana se utiliza el botón cancel, retornando a la ventana principal del sistema DOMINIUS-BP.

E. Anexo: Tablas de valores de error para los servicios descritos en el capítulo 6

Este anexo, presenta las tablas en las que se asocia los valores de error para cada servicio descrito en el capítulo 6 de esta tesis, en los días martes 17pm y jueves 11am y 5pm. (**Tabla E-1, Tabla E-2, Tabla E-3, Tabla E-4, Tabla E-5, Tabla E-6, Tabla E-7, Tabla E-8, Tabla E-9, Tabla E-10, Tabla E-11, Tabla E-12**)**.**

Tabla E-1: Error de los servicios buyItem calculado el día martes a las 5pm con el criterio de disponibilidad.

MARTES 5 p.m. (Disponibilidad)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.0999	0.1236	0.1158	0.1218	0.0832
0.0689	0.085	0.0796	0.0837	0.0576
0.0558	0.0688	0.0645	0.0678	0.0466
0.0481	0.0593	0.0556	0.0584	0.0402
0.0429	0.0529	0.0496	0.0521	0.0359
0.0391	0.0482	0.0452	0.0475	0.0327
0.0362	0.0445	0.0418	0.0439	0.0302
0.0339	0.0416	0.0391	0.0409	0.0282
0.0318	0.0393	0.0368	0.0387	0.0266
0.0302	0.0372	0.0349	0.0367	0.0253
0.0288	0.0355	0.0332	0.0349	0.0241
0.0276	0.0339	0.0319	0.0333	0.023
0.0265	0.0326	0.0306	0.0321	0.0221
0.0256	0.0313	0.0295	0.0308	0.0213
0.0246	0.0303	0.0284	0.0299	0.0206

Tabla E-2: Error de los servicios getItemInformation calculado el día martes a las 5pm con el criterio de disponibilidad.

MARTES 5 p.m. (Disponibilidad)				
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.1202	0.1228	0.1116	0.1238	0.0898
0.0826	0.0844	0.0768	0.0851	0.062
0.0669	0.0683	0.0622	0.0689	0.0502
0.0576	0.0589	0.0535	0.0594	0.0432
0.0514	0.0526	0.0478	0.053	0.0387
0.0469	0.0479	0.0436	0.0483	0.0352
0.0433	0.0443	0.0402	0.0446	0.0324
0.0404	0.0414	0.0375	0.0417	0.0302
0.0382	0.039	0.0355	0.0393	0.0287
0.0362	0.037	0.0336	0.0373	0.0272
0.0345	0.0352	0.0321	0.0355	0.0259
0.0329	0.0337	0.0305	0.0339	0.0246
0.0317	0.0324	0.0294	0.0326	0.0238
0.0304	0.0312	0.0281	0.0314	0.0226
0.0295	0.0301	0.0274	0.0304	0.0222

Tabla E-3: Error de los servicios buyItem calculado el día martes a las 5pm con restricción de 1000 ms, con el criterio de reactividad

MARTES 5 p.m. (Reactividad 1000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.1232	0.1232	0.0947	0.1222	0.1215
0.0847	0.0847	0.0653	0.0841	0.0836
0.0686	0.0686	0.0529	0.068	0.0677
0.0591	0.0591	0.0455	0.0586	0.0583
0.0528	0.0528	0.0407	0.0523	0.0521
0.0481	0.0481	0.0371	0.0477	0.0474
0.0444	0.0444	0.0342	0.0441	0.0438
0.0415	0.0415	0.0318	0.0412	0.041
0.0391	0.0391	0.0302	0.0388	0.0386
0.0371	0.0371	0.0286	0.0368	0.0366
0.0354	0.0354	0.0273	0.0351	0.0349
0.0338	0.0338	0.0259	0.0335	0.0334
0.0325	0.0325	0.025	0.0322	0.0321

MARTES 5 p.m. (Reactividad 1000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.0313	0.0313	0.0238	0.031	0.0309
0.0303	0.0303	0.0234	0.03	0.0298

Tabla E-4: Error de los servicios getItemInformation calculado el día martes a las 5pm con restricción de 1000 ms, con el criterio de reactividad

MARTES 5 p.m. (Reactividad 1000 milisegundos)				
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.1158	0.1218	0.0832	0.1229	0.1084
0.0796	0.0837	0.0576	0.0845	0.0747
0.0645	0.0678	0.0466	0.0684	0.0604
0.0555	0.0584	0.0401	0.0589	0.052
0.0496	0.0521	0.0359	0.0526	0.0465
0.0452	0.0475	0.0327	0.0479	0.0424
0.0417	0.0439	0.0301	0.0443	0.0391
0.0389	0.0409	0.028	0.0413	0.0364
0.0368	0.0387	0.0266	0.039	0.0345
0.0349	0.0367	0.0253	0.037	0.0327
0.0332	0.0349	0.0241	0.0353	0.0312
0.0317	0.0333	0.0228	0.0337	0.0297
0.0305	0.0321	0.0221	0.0324	0.0286
0.0292	0.0308	0.021	0.0311	0.0273
0.0284	0.0299	0.0206	0.0302	0.0267

Tabla E-5: Error de los servicios buyItem calculado el día martes a las 5pm con restricción de 5000 ms, con el criterio de reactividad

MARTES 5 p.m. (Reactividad 5000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.0832	0.0855	0.0843	0.0883	0.0822
0.0576	0.0591	0.0583	0.061	0.0569
0.0466	0.0479	0.0472	0.0494	0.0461
0.0402	0.0412	0.0406	0.0425	0.0397
0.0359	0.0369	0.0364	0.038	0.0355
0.0327	0.0336	0.0331	0.0346	0.0323
0.0302	0.0309	0.0305	0.0319	0.0299

MARTES 5 p.m. (Reactividad 5000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.0282	0.0288	0.0284	0.0297	0.0279
0.0266	0.0274	0.027	0.0282	0.0263
0.0253	0.0259	0.0256	0.0268	0.025
0.0241	0.0247	0.0244	0.0255	0.0238
0.023	0.0234	0.0231	0.0242	0.0227
0.0221	0.0227	0.0223	0.0234	0.0219
0.0213	0.0216	0.0213	0.0222	0.021
0.0206	0.0211	0.0209	0.0218	0.0204

Tabla E-6: Error de los servicios getItemInformation calculado el día martes a las 5pm con restricción de 5000 ms, con el criterio de reactividad

MARTES 5 p.m. (Reactividad 5000 milisegundos)				
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.0913	0.0981	0.0802	0.0883	0.0913
0.0631	0.0677	0.0556	0.061	0.0631
0.0511	0.0548	0.045	0.0494	0.0511
0.0439	0.0471	0.0387	0.0425	0.0439
0.0393	0.0422	0.0347	0.038	0.0393
0.0358	0.0384	0.0316	0.0346	0.0358
0.033	0.0354	0.0291	0.0319	0.033
0.0307	0.033	0.0271	0.0297	0.0307
0.0292	0.0313	0.0257	0.0282	0.0292
0.0277	0.0297	0.0244	0.0268	0.0277
0.0264	0.0283	0.0232	0.0255	0.0264
0.025	0.0269	0.0221	0.0242	0.025
0.0242	0.0259	0.0213	0.0234	0.0242
0.023	0.0247	0.0203	0.0222	0.023
0.0226	0.0242	0.0199	0.0218	0.0226

Tabla E-7: Error de los servicios buyItem calculado el día martes a las 5pm con restricción de de 10000 ms, con el criterio de reactividad

MARTES 5 p.m. (Reactividad 10000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.0898	0.0807	0.121	0.0797	0.1182

MARTES 5 p.m. (Reactividad 10000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.062	0.0559	0.0832	0.0552	0.0813
0.0502	0.0453	0.0674	0.0447	0.0658
0.0433	0.039	0.0581	0.0385	0.0568
0.0387	0.0349	0.0518	0.0344	0.0506
0.0352	0.0318	0.0472	0.0314	0.0461
0.0326	0.0293	0.0437	0.0289	0.0427
0.0305	0.0272	0.0408	0.0269	0.0399
0.0287	0.0259	0.0384	0.0256	0.0375
0.0272	0.0245	0.0364	0.0242	0.0356
0.0259	0.0234	0.0347	0.0231	0.0339
0.0248	0.0222	0.0333	0.022	0.0325
0.0238	0.0214	0.0319	0.0212	0.0312
0.023	0.0204	0.0308	0.0202	0.0301
0.0222	0.02	0.0297	0.0198	0.029

Tabla E-8: Error de los servicios getItemInformation calculado el día martes a las 5pm con reactividad de 10000 ms, con el criterio de reactividad

MARTES 5 p.m. (Reactividad 10000 milisegundos)				
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.0822	0.0883	0.1068	0.0814	0.0913
0.0569	0.061	0.0735	0.0564	0.0631
0.0461	0.0494	0.0595	0.0457	0.0511
0.0397	0.0425	0.0514	0.0393	0.0441
0.0355	0.038	0.0458	0.0351	0.0393
0.0323	0.0346	0.0417	0.032	0.0358
0.0299	0.0319	0.0387	0.0296	0.0332
0.0279	0.0297	0.0362	0.0276	0.031
0.0263	0.0282	0.034	0.0261	0.0292
0.025	0.0268	0.0322	0.0247	0.0277
0.0238	0.0255	0.0307	0.0236	0.0264
0.0227	0.0242	0.0295	0.0225	0.0253
0.0219	0.0234	0.0283	0.0216	0.0242
0.021	0.0222	0.0273	0.0208	0.0234
0.0204	0.0218	0.0263	0.0202	0.0226

Tabla E-9: Error de los servicios buyItem y getItemInformation calculado el día jueves a las 11am, con el criterio de disponibilidad

JUEVES 11 a.m. (Disponibilidad)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.1116	0.1237	0.093	0.1237	0.0981
0.0768	0.0851	0.0642	0.0851	0.0677
0.0622	0.0689	0.052	0.0689	0.0548
0.0537	0.0593	0.0449	0.0593	0.0471
0.0478	0.053	0.04	0.053	0.0422
0.0436	0.0483	0.0365	0.0483	0.0384
0.0404	0.0446	0.0338	0.0446	0.0354
0.0378	0.0416	0.0316	0.0416	0.033
0.0355	0.0393	0.0297	0.0393	0.0313
0.0336	0.0373	0.0282	0.0373	0.0297
0.0321	0.0355	0.0268	0.0355	0.0283
0.0308	0.0339	0.0257	0.0339	0.0269
0.0295	0.0326	0.0247	0.0326	0.0259
0.0285	0.0314	0.0238	0.0314	0.0247
0.0274	0.0304	0.0229	0.0304	0.0242
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.113	0.1236	0.11	0.0947	0.1233
0.0778	0.085	0.0757	0.0653	0.0848
0.063	0.0688	0.0613	0.0529	0.0686
0.0543	0.0593	0.0529	0.0455	0.0591
0.0484	0.0529	0.0472	0.0407	0.0528
0.0441	0.0482	0.043	0.0371	0.0481
0.0409	0.0445	0.0398	0.0342	0.0444
0.0382	0.0416	0.0373	0.0318	0.0415
0.0359	0.0393	0.035	0.0302	0.0392
0.0341	0.0372	0.0332	0.0286	0.0371
0.0325	0.0355	0.0316	0.0273	0.0354
0.0312	0.0339	0.0304	0.0259	0.0338
0.0299	0.0326	0.0291	0.025	0.0325
0.0289	0.0313	0.0281	0.0238	0.0312
0.0278	0.0303	0.027	0.0234	0.0303

Tabla E-10: Error de los servicios buyItem y getItemInformation calculado el día jueves a las 11am, con restricción de 1000ms con el criterio de reactividad

JUEVES 11 a.m. (Reactividad 1000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.1237	0.0964	0.0802	0.1232	0.1182
0.0851	0.0665	0.0556	0.0847	0.0813
0.0689	0.0539	0.045	0.0686	0.0658
0.0593	0.0463	0.0387	0.0591	0.0566
0.053	0.0414	0.0347	0.0528	0.0506
0.0483	0.0378	0.0316	0.0481	0.0461
0.0446	0.0348	0.0291	0.0444	0.0426
0.0416	0.0324	0.0271	0.0415	0.0397
0.0393	0.0307	0.0257	0.0391	0.0375
0.0373	0.0292	0.0244	0.0371	0.0356
0.0355	0.0278	0.0232	0.0354	0.0339
0.0339	0.0264	0.0221	0.0338	0.0323
0.0326	0.0255	0.0213	0.0325	0.0311
0.0314	0.0243	0.0203	0.0313	0.0298
0.0304	0.0238	0.0199	0.0303	0.029
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.0883	0.0913	0.0799	0.1218	0.0797
0.061	0.0631	0.0553	0.0837	0.0552
0.0494	0.0511	0.0448	0.0678	0.0447
0.0425	0.0439	0.0386	0.0584	0.0385
0.038	0.0393	0.0345	0.0521	0.0344
0.0346	0.0358	0.0315	0.0475	0.0314
0.0319	0.033	0.029	0.0439	0.0289
0.0297	0.0307	0.027	0.0409	0.027
0.0282	0.0292	0.0256	0.0387	0.0256
0.0268	0.0277	0.0243	0.0367	0.0242
0.0255	0.0264	0.0231	0.0349	0.0231
0.0242	0.025	0.022	0.0333	0.022
0.0234	0.0242	0.0212	0.0321	0.0212
0.0222	0.023	0.0202	0.0308	0.0202
0.0218	0.0226	0.0198	0.0299	0.0198

Tabla E-11: Error de los servicios buyItem y getItemInformation calculado el día jueves a las 11am, con restricción de 5000ms con el criterio de reactividad

JUEVES 11 a.m. (Reactividad 5000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.0999	0.0807	0.0802	0.0822	0.0855
0.0689	0.0559	0.0556	0.0569	0.0591
0.0558	0.0453	0.045	0.0461	0.0479
0.048	0.039	0.0387	0.0397	0.0413
0.0429	0.0349	0.0347	0.0355	0.0369
0.0391	0.0318	0.0316	0.0323	0.0336
0.036	0.0293	0.0291	0.0299	0.0311
0.0335	0.0272	0.0271	0.0279	0.029
0.0318	0.0259	0.0257	0.0263	0.0274
0.0302	0.0245	0.0244	0.025	0.0259
0.0288	0.0234	0.0232	0.0238	0.0247
0.0273	0.0222	0.0221	0.0227	0.0237
0.0264	0.0214	0.0213	0.0219	0.0227
0.0252	0.0204	0.0203	0.021	0.0219
0.0246	0.02	0.0199	0.0204	0.0211
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.0868	0.0797	0.0797	0.0822	0.0947
0.06	0.0552	0.0552	0.0569	0.0653
0.0486	0.0447	0.0447	0.0461	0.0529
0.0419	0.0385	0.0385	0.0397	0.0457
0.0374	0.0344	0.0344	0.0355	0.0407
0.0341	0.0314	0.0314	0.0323	0.0371
0.0315	0.0289	0.0289	0.0298	0.0344
0.0295	0.0269	0.027	0.0277	0.0321
0.0278	0.0256	0.0256	0.0263	0.0302
0.0263	0.0242	0.0242	0.025	0.0287
0.0251	0.0231	0.0231	0.0238	0.0273
0.024	0.022	0.022	0.0226	0.0262
0.0231	0.0212	0.0212	0.0218	0.0251
0.0222	0.0202	0.0202	0.0208	0.0243
0.0215	0.0198	0.0198	0.0204	0.0234

Tabla E-12: Error de los servicios buyItem y getItemInformation calculado el día jueves a las 11am, con restricción de 10000ms con el criterio de reactividad

JUEVES 11 a.m. (Reactividad 10000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.093	0.1182	0.1158	0.0981	0.1145
0.0642	0.0813	0.0796	0.0677	0.0787
0.052	0.0658	0.0645	0.0548	0.0637
0.0449	0.0568	0.0556	0.0473	0.055
0.04	0.0506	0.0496	0.0422	0.049
0.0365	0.0461	0.0452	0.0384	0.0447
0.0338	0.0427	0.0418	0.0356	0.0414
0.0316	0.0399	0.0391	0.0333	0.0387
0.0297	0.0375	0.0368	0.0313	0.0364
0.0282	0.0356	0.0349	0.0297	0.0345
0.0268	0.0339	0.0332	0.0283	0.0329
0.0257	0.0325	0.0319	0.0271	0.0315
0.0247	0.0312	0.0306	0.026	0.0302
0.0238	0.0301	0.0295	0.0252	0.0292
0.0229	0.029	0.0284	0.0242	0.0281
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.0964	0.0913	0.0797	0.0855	0.1016
0.0665	0.0631	0.0552	0.0591	0.0701
0.0539	0.0511	0.0447	0.0479	0.0567
0.0465	0.0441	0.0385	0.0413	0.049
0.0414	0.0393	0.0344	0.0369	0.0436
0.0378	0.0358	0.0314	0.0336	0.0398
0.035	0.0332	0.0289	0.0311	0.0368
0.0327	0.031	0.027	0.029	0.0345
0.0307	0.0292	0.0256	0.0274	0.0324
0.0292	0.0277	0.0242	0.0259	0.0307
0.0278	0.0264	0.0231	0.0247	0.0293
0.0267	0.0253	0.022	0.0237	0.0281
0.0256	0.0242	0.0212	0.0227	0.0269
0.0247	0.0234	0.0202	0.0219	0.026
0.0238	0.0226	0.0198	0.0211	0.025

Tabla E-13: Error de los servicios buyItem y getItemInformation calculado el día jueves a las 5pm con el criterio de disponibilidad.

JUEVES 5 pm. (Disponibilidad)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.0898	0.1228	0.113	0.1229	0.1145
0.062	0.0844	0.0778	0.0845	0.0787
0.0502	0.0683	0.063	0.0684	0.0637
0.0432	0.0589	0.0543	0.0589	0.055
0.0387	0.0526	0.0484	0.0526	0.049
0.0352	0.0479	0.0441	0.0479	0.0447
0.0324	0.0443	0.0409	0.0443	0.0414
0.0302	0.0414	0.0382	0.0413	0.0387
0.0287	0.039	0.0359	0.039	0.0364
0.0272	0.037	0.0341	0.037	0.0345
0.0259	0.0352	0.0325	0.0353	0.0329
0.0246	0.0337	0.0312	0.0337	0.0315
0.0238	0.0324	0.0299	0.0324	0.0302
0.0226	0.0312	0.0289	0.0311	0.0292
0.0222	0.0301	0.0278	0.0302	0.0281
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.1051	0.1237	0.0799	0.0814	0.1235
0.0724	0.0851	0.0553	0.0564	0.085
0.0586	0.0689	0.0448	0.0457	0.0688
0.0506	0.0593	0.0386	0.0393	0.0593
0.0451	0.053	0.0345	0.0351	0.0529
0.0411	0.0483	0.0315	0.032	0.0482
0.0381	0.0446	0.029	0.0296	0.0445
0.0356	0.0416	0.027	0.0276	0.0416
0.0335	0.0393	0.0256	0.0261	0.0392
0.0317	0.0373	0.0243	0.0247	0.0372
0.0302	0.0355	0.0231	0.0236	0.0355
0.029	0.0339	0.022	0.0225	0.0339
0.0278	0.0326	0.0212	0.0216	0.0326
0.0269	0.0314	0.0202	0.0208	0.0313
0.0259	0.0304	0.0198	0.0202	0.0303

Tabla E-14: Error de los servicios buyItem y getItemInformation calculado el día jueves a las 5pm con restricción de 1000ms, con el criterio de reactividad.

JUEVES 5 pm. (Reactividad 1000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.1232	0.1229	0.0832	0.1192	0.0868
0.0847	0.0845	0.0576	0.082	0.06
0.0686	0.0684	0.0466	0.0664	0.0486
0.0591	0.0589	0.0401	0.0571	0.0418
0.0528	0.0526	0.0359	0.051	0.0374
0.0481	0.0479	0.0327	0.0465	0.0341
0.0444	0.0443	0.0301	0.0429	0.0314
0.0415	0.0413	0.028	0.0401	0.0292
0.0391	0.039	0.0266	0.0379	0.0278
0.0371	0.037	0.0253	0.0359	0.0263
0.0354	0.0353	0.0241	0.0342	0.0251
0.0338	0.0337	0.0228	0.0326	0.0238
0.0325	0.0324	0.0221	0.0314	0.023
0.0313	0.0311	0.021	0.0301	0.0219
0.0303	0.0302	0.0206	0.0293	0.0215
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.1202	0.1051	0.1236	0.0802	0.1068
0.0826	0.0724	0.085	0.0556	0.0735
0.0669	0.0586	0.0688	0.045	0.0595
0.0576	0.0504	0.0593	0.0387	0.0512
0.0514	0.0451	0.0529	0.0347	0.0458
0.0469	0.0411	0.0482	0.0316	0.0417
0.0433	0.0379	0.0445	0.0291	0.0385
0.0404	0.0353	0.0416	0.0271	0.0358
0.0382	0.0335	0.0393	0.0257	0.034
0.0362	0.0317	0.0372	0.0244	0.0322
0.0345	0.0302	0.0355	0.0232	0.0307
0.0329	0.0287	0.0339	0.0221	0.0292
0.0317	0.0277	0.0326	0.0213	0.0282
0.0304	0.0265	0.0313	0.0203	0.0269
0.0295	0.0259	0.0303	0.0199	0.0263

Tabla E-15: Error de los servicios buyItem y getItemInformation calculado el día jueves a las 5pm con restricción de 5000ms, con el criterio de reactividad.

JUEVES 5 pm. (Reactividad 5000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.0898	0.093	0.0913	0.0981	0.0807
0.062	0.0642	0.0631	0.0677	0.0559
0.0502	0.052	0.0511	0.0548	0.0453
0.0432	0.0447	0.0441	0.0471	0.039
0.0387	0.04	0.0393	0.0422	0.0349
0.0352	0.0365	0.0358	0.0384	0.0318
0.0324	0.0336	0.0332	0.0354	0.0293
0.0302	0.0312	0.031	0.033	0.0274
0.0287	0.0297	0.0292	0.0313	0.0259
0.0272	0.0281	0.0277	0.0297	0.0245
0.0259	0.0268	0.0264	0.0283	0.0234
0.0246	0.0255	0.0253	0.0269	0.0223
0.0238	0.0246	0.0242	0.0259	0.0215
0.0226	0.0234	0.0234	0.0247	0.0206
0.0222	0.0229	0.0226	0.0242	0.02
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.1016	0.0807	0.1182	0.0797	0.0981
0.0701	0.0559	0.0813	0.0552	0.0677
0.0567	0.0453	0.0658	0.0447	0.0548
0.0488	0.039	0.0566	0.0385	0.0471
0.0436	0.0349	0.0506	0.0344	0.0422
0.0398	0.0318	0.0461	0.0314	0.0384
0.0366	0.0293	0.0426	0.0289	0.0354
0.0341	0.0272	0.0397	0.0269	0.033
0.0324	0.0259	0.0375	0.0256	0.0313
0.0307	0.0245	0.0356	0.0242	0.0297
0.0293	0.0234	0.0339	0.0231	0.0283
0.0278	0.0222	0.0323	0.022	0.0269
0.0268	0.0214	0.0311	0.0212	0.0259
0.0256	0.0204	0.0298	0.0202	0.0247
0.025	0.02	0.029	0.0198	0.0242

Tabla E-16: Error de los servicios buyItem y getItemInformation calculado el día jueves a las 5pm con restricción de 10000ms, con el criterio de reactividad.

JUEVES 5 pm. (Reactividad 10000 milisegundos)				
BuyItem001	**BuyItem002**	**BuyItem003**	**BuyItem004**	**BuyItem005**
0.0807	0.0883	0.1068	0.0964	0.0981
0.0559	0.061	0.0735	0.0665	0.0677
0.0453	0.0494	0.0595	0.0539	0.0548
0.039	0.0425	0.0514	0.0463	0.0473
0.0349	0.038	0.0458	0.0414	0.0422
0.0318	0.0346	0.0417	0.0378	0.0384
0.0293	0.0319	0.0387	0.0348	0.0356
0.0274	0.0297	0.0362	0.0324	0.0333
0.0259	0.0282	0.034	0.0307	0.0313
0.0245	0.0268	0.0322	0.0292	0.0297
0.0234	0.0255	0.0307	0.0278	0.0283
0.0223	0.0242	0.0295	0.0264	0.0271
0.0215	0.0234	0.0283	0.0255	0.026
0.0206	0.0222	0.0273	0.0243	0.0252
0.02	0.0218	0.0263	0.0238	0.0242
GetItemInformation001	**GetItemInformation002**	**GetItemInformation003**	**GetItemInformation004**	**GetItemInformation005**
0.0843	0.0822	0.0981	0.0799	0.0883
0.0583	0.0569	0.0677	0.0553	0.061
0.0472	0.0461	0.0548	0.0448	0.0494
0.0406	0.0397	0.0471	0.0386	0.0425
0.0364	0.0355	0.0422	0.0345	0.038
0.0331	0.0323	0.0384	0.0315	0.0346
0.0305	0.0299	0.0354	0.029	0.0319
0.0284	0.0279	0.033	0.027	0.0297
0.027	0.0263	0.0313	0.0256	0.0282
0.0256	0.025	0.0297	0.0243	0.0268
0.0244	0.0238	0.0283	0.0231	0.0255
0.0231	0.0227	0.0269	0.022	0.0242
0.0223	0.0219	0.0259	0.0212	0.0234
0.0213	0.021	0.0247	0.0203	0.0222
0.0209	0.0204	0.0242	0.0198	0.0218

Glosario

Aprendizaje de máquinas: rama de la inteligencia artificial que le permite a las máquinas desarrollar técnicas que a partir de información no estructuradas generalizar comportamientos.

Árboles de decisión: técnica de predicción utilizada en la inteligencia artificial para adquirir comportamientos a partir de un conjunto de información.

Árboles de decisión lógicos: técnica de aprendizaje automático que asocia la descripción de los árboles de decisión y la programación lógica inductiva.

Calidad de servicios (QoS): conjunto de aspectos que permiten calificar el comportamiento de los servicios.

Composición de servicios Web: método utilizado para enlazar servicios Web distribuidos en la Web

Entropía: método estadístico para calcular la incertidumbre de un conjunto de datos.

Error cuadrático medio: método estadístico para cuantificar la diferencia entre los valores de un estimador y los verdaderos valores estimados.

Manejo de riesgos: es la identificación, evaluación y priorización de los riesgos

Programación lógica inductiva: técnica de la inteligencia artificial para aprender reglas de la lógica de primer orden que describan un predicado a partir de un conocimiento base y de un conjunto de ejemplos, denominado conjunto de entrenamiento, donde puede haber tanto ejemplos positivos como negativos.

Rangos recíprocos (RR): método estadístico utilizado para calcular el impacto de un conjunto de variables

Riesgo: Es la posibilidad de que un servicio seleccionado dará lugar a una falla

Servicio Web: conjunto de aplicaciones modulares, autónomas, auto-descritas que se encuentran publicadas, localizadas e invocadas a través de la Web y, por medio del intercambio de sus datos, proporcionan un conjunto de servicios.

Bibliografía

Aamodt, A., & Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications , 7* (1), 39-59.

Abramovich, F., & Angelini, C. (2006). Bayesian Maximum a Posteriori Multiple Testing Procedure. *68*, 436-460.

Ahmed, N., & Gokhal, D. V. (1989). Entropy expressions and their estimators for multivariate distribution. *IEEE Transactions on Information Theory* , 688-692.

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A., y otros. (2005). *Web Service Semantics - WSDL-S.* A joint UGA-IBM Technical Note, IBM.

Araujo Sierra, B. (2006). *Aprendizaje Automatico: conceptos básicos y avanzados.* (P. Hall, Ed.) Pearson.

Banerji, A., Bartolini, C., Beringe, D., Chopella, V., Govindarajan, K., Karp, A., y otros. (2002). *Web Services Conversation Language (WSCL) 1.0.* Note, W3C.

Barreiro, D., Albers, P., & Hao, J.-K. (2006). Web Services Composition. En J. Cardoso, & A. P. Sheth, *Semantic Web Services, Processes and Applications* (págs. 195-225). New York: SpringerLink.

Barto, A., & Duff, M. (1994). Monte Carlo matrix inversion and reinforcement learning. *In Advances in Neural Information Processing Systems , 6*, 687-694.

Berardi, D. (2004). *Automatic Service Composition: Models, Techniques and Tools.* Final Ph.D Thesis., Universidad La Sapienza.

Blockeel, H., & Raedt, L. D. (1998). Top-down induction of first order logical decision trees. *Artificial Intelligence , 101* (1-2), 285-297.

Brogi, A., Canal, C., Pimentel, E., & Vallecillo, A. (2004). Formalizing Web Service Choreographies. *Journal Electronic Notes in Theoretical Computer Science (ENTCS)* , 73-94.

Busoniu, L., Babuska, R., Schutter, B. D., & Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators.* CRC Press, Automation and Control Engineering Series.

Bussler, C., Hull, R., McIlraith, S., Orlowska, M., Pernici, B., & Yang, J. (2002). Web Services, E-Business, and the Semantic Web. *CAiSE 2002 International Workshop, WES 2002* .

Cardinale, Y., El Haddad, J., Manouvrier, M., & Rukoz, M. (2010). Web Service Selection for Transactional Composition. *International Conference on Computational Science, ICCS 2010* , 2683 - 2692.

Cardoso, J., Sheth, A., Miller, J., Arnold, J., & Kochut, K. (2004). Quality of Service for Workflows and Web Service Processes. *Journal of Web Semantics* .

Carman, M., & Knoblock, C. (2007). Learning Semantic Descriptions of Web Information Sources. *Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)* , 2695-2700.

Carman, M., Serafini, L., & Traverso, P. (2003). Web Service Composition as Planning. *In ICAPS 2003 Workshop on Planning for Web Services* .

Chakraborty, D., Perich, F., Joshi, A., Finin, T., & Yesha, Y. (2002). A Reactive Service Composition Architecture for Pervasive Computing Environments. *In: 7th Personal Wireless Communications Conference* .

Chan, M., & Bishop, J. (2009). The Design of a self-Healing Composition Cycle for Web Services. *seams, 2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems* , 20-27.

Chan, M., Bishop, J., Steyn, J., Baresi, L., & Guinea, S. (2009). A Fault Taxonomy for Web Service Composition. En *Service-Oriented Computing - ICSOC 2007 Workshops* (págs. 363-37). Berlin: Springer-Verlag Berlin, Heidelber.

Christensen, E., Curbera, F., Meredith, G., & Weerawarana, S. (2001). *Web services description language (WSDL) 1.1.* Note, W3C.

Curbera, F., Khalaf, R., Mukhi, N., Tai, S., & Weerawarana, S. (2003). The next step in Web services. *Communications of the ACM - Service-oriented computing , 46*, 29-34.

Dietterich, T. (2003). Machine Learning. En *Nature Encyclopedia of Cognitive Science.* London: Macmillan.

Ebbers, M., Williams, N., Lopez, L., Herman, R., Able, G., Chieregatti, P., y otros. (2004). *Implementing CICS Web Services.* Redbooks.

El Haddad, J., Manouvrier, M., Ramirez, G., & Rukoz, M. (2008). QoS-driven Selection ofWeb Services for Transactional Composition. *IEEE International Conference on Web Services* , 653-660 .

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated Planning: Theory & Practice.* Morgan Kaufmann.

Gosavi, A. (2004). A Reinforcement Learning Algorithm Based on Policy Iteration for Average Reward: Empirical Results with Yield Management and Convergence Analysis. En *MACHINE LEARNING* (Vol. 55, págs. 5-29). Springer.

Gunther, N. (2000). *The Practical Performance Analyst.* iUniverse Press.

Guzman, J., & Ovalle, D. (2008b). INDIGO: Una Propuesta de Planificación en Inteligencia Artificial para la Composición Automática de Servicios Web Semánticos. *VII Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC '08) , 7* (1), 51-57.

Guzman, J., & Ovalle, D. (2009). Reactive Planning Model for Web Service Composition under Incomplete Information and Time Restrictions. *Proceedings of the 2009 Third International Conference on Digital Society* , 196-201.

Guzman, J., & Ovalle, D. (2008). Web Services Planning Agent in Dynamic Environments with Incomplete Information and Time Restrictions. *The 11th IEEE Internacional Conference CSE 2008* , 245-250.

Juric, M., Mathew, B., & Sarang, P. (2004). *Business Process Execution Language for Web Services : BPEL and BPEL4WS.* PACKT publishing.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research , 4*, 237-285.

Karalic, A. (1992). Employing linear regression in regression tree leaves. *Proceedings of the 10th European conference on Artificial intelligence. ECAI 1992* , 440-441.

Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., & Barreto, C. (2005). *Web services choreography description language version 1.0.* Candidate recommendation, W3C.

Kersting, K. (2006). *An Inductive Logic Programming Approach To Statistical Relational Learning.* (I. Press, Ed.)

Kokash, N. (2008). *Engineering Service-Oriented Systems: Modeling, Discovery and Quality.* Italia.

Kokash, N., & D'Andrea, V. (2007). Evaluating Quality of Web Services: A Risk-Driven Approach. En *Business Information Systems* (págs. 180-194).

Kolodner, J. (1993). *Case Based Reasoning.* (M. Kaufmann, Ed.)

Kolodner, J. (1983). Reconstructive Memory: A Computer Model. *Cognitive Science , 7* (4), 281-328.

Lavrac, N., & Saso, D. (1994). *Inductive Logic Programming Techniques and Applications.* (E. Horwood, Ed.) New York.

Lee, S.-Y., & Song, X.-Y. (2003). Maximum Likelihood Estimation and Model Comparison for Mixtures of Structural Equation Models with Ignorable Missing Data. *20* (2), 221-255.

Li, Z., Su, S., & Yang, F. (2007). WSrep: A Novel Reputation Model for Web Services Selection. *Lecture Notes in Computer Science , 4496*, 199-208.

Liu, G.-q., Zhu, Z.-l., Li, Y.-q., Li, D.-c., & Cui, J.-c. (2009). A New Web Service Model Based On QoS. *International Symposium on Intelligent Ubiquitous Computing and Education* , 395-399.

Liu, H., Zhang, W., Ren, K., Liu, C., & Zhang, Z. (2009). A Risk-driven Selection Approach for Transactional Web Service Composition. *Eighth International Conference on Grid and Cooperative Computing* , 391-397.

Liu, Y.-Z., QIU, S., TAO, H.-J., ZANG, T.-Y., & WANG, Y.-D. (2009). A CASE-BASED REASONING APPROACH TO SUPPORT WEB SERVICE COMPOSITION. *Proceedings of the Eighth International Conference on Machine Learning and Cybernetics* , 12-15.

Mani, A., & Nagarajan, A. (2002). *Understanding quality of service for Web services.* Technical report, IBM.

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., y otros. (2004). Bringing Semantics to Web Services: An Approach OWL-S. *In SWSWPC 2004 , 3387*, 26-42.

McIlraith, S., Son, T., & Zeng, H. (2001). Semantic Web Services. *IEEE Intelligent Systems , 16* (2), 46–53.

Mikic-Rakic, M., Malek, S., & Medvidovic, N. (2005). Improving Availability in Large, Distributed Component-Based Systems Via Redeployment. *In proceedings of the 3rd International Conference on Component Deployment (CD 2005)* .

Mitchell, T. (1997). *Machine Learning.* McGraw-Hill.

Mood, A., Graybill, F., & Boes, D. (1974). *Introduction to the Theory of Statistics.* McGraw-Hill.

Muggleton, S. (1991). Inductive logic programming. *New Generation Computing , 8* (4), 295-318.

Muggleton, S. (1992). Inverting Implication. *Artificial Intelligence Journal* .

Muggleton, S., & Buntine, W. (1988). Machine Invention of First-Order Predicates by. Inverting Resolution. (M. Kaufmann, Ed.) *In: Proceedings of the 5th International Conference on Machine Learning ICML'88Morgan Kaufmann (1988)* , 339--352.

Muggleton, S., & Raedt, L. D. (1994). Inductive Logic Programming: Theory and Methods. *Academic Press , 19*, 629-679.

Nadalin, A., Kaler, C., Monzillo, R., & Hallam-Baker, P. (2006). *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004).* Technical report, OASIS Web Service Security (WSS).

Nam, W., Hyunyoung, K., & Lee, J. (2009). QoS-DrivenWeb Service Composition Using Learning-based Depth First Search. *2009 IEEE Conference on Commerce and Enterprise Computing* , 507-510.

Nurm, D., Brevik, J., & Wolski, R. (2005). Modeling Machine Availability in Enterprise and Wide-area Distributed Computing Environments. *EURO-PAR 2005 PARALLEL PROCESSING , 3648*, 432-441.

Peltz, C. (2003). Web services orchestration and choreography. *Hewlett-Packard Company. Publicado por IEEE Computer Society.*

Pereira, A., Franco, G., Silva, L., Meira, W., & Santos, W. (2004). The USAR characterization model. *In Proceedings of the IEEE 7th Workshop on Workload Characterization* , 63-70.

Plotkin, G. (1969). A Note on Inductive Generalization. *In: Machine Learning , 5*, 153-163.

Poole, D. (1993). Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence , 64* (1), 81-129.

Qian, S., Lu, S., & Lie, X. (2005). Mobile Agent Based Web Services Composition. *Springer Berlin / Heidelberg , 3795*, 35-46.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning.* (M. K. Inc., Ed.) San Francisco, USA.

Qulnlan, J. (1986). Induction of decision trees. *Machine Learning , 1* (1), 81-106.

Ran, S. (2003). A Model for Web Services Discovery With QoS. *ACM SIGecom Exchanges , 4* (1), 1-10 .

Rao, J., & Su, X. (2005). A Survey of Automated Web Service Composition Methods. *Semantic Web Services and Web Process Composition* , 45-54.

Robinson, A. (1965). A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM , 12* (1), 23-41.

Roman, D., Keller, U., Lausen, H., Bruijn, J. d., Lara, R., Stollberg, M., y otros. (2005). Web Service Modeling Ontology. *Applied Ontology , 1* (1), 77 - 106.

Rosenberg, F., Platzer, C., & Dus, S. (2006). Bootstrapping Performance and Dependability Attributes of Web Services. *International conference on Web Services* , 205-212.

Russell, S., & Norving, P. (2004). *Inteligencia Artificial Un Enfoque Práctico, 2da. Edición.* (P. Hall, Ed.) Pearson.

Saeid, M., Abd Ghani, A. A., & Selamat, H. (2011). Rank-Order Weighting of Web Attributes for Website Evaluation. *The International Arab Journal of Information Technology* , 30-38.

Sebe, N., Cohen, I., Garg, A., & H., T. S. (2005). *Machine Learning in Computer Vision.* (Springer, Ed.)

Snell, J., Tidwell, D., & Kulchenko, P. (2001). *Programming Web Services with SOAP.* O'Reilly Media.

Strehl, A., Li, L., & Littman, M. (2006). Incremental Model-based Learners With Formal Learning-Time Guarantees. *In Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence.*

Strehl, A., Li, L., Wiewiora, E., Langford, J., & Littman, M. (2006). Pac model-free reinforcement learning. *In: ICML-06: Proceedings of the 23rd international conference on Machine learning.*

Swinburne, R. (2005). *Bayes's Theorem.* Proceedings of the British Academy.

Triantaphyllou, E. (2000). *Multi-Criteria Decision Making Methods: A Comparative Study* (Vol. 44). Florida: Kluwer Academi Publishers.

Verdon, D., & McGraw, G. (2004). Risk analysis in software design. *In: IEEE Security and Privacy , 2* (4), 79-84.

Walsh, A. (2002). *UDDI, SOAP, and WSDL: The Web Services Specification Reference Book.* Prentice Hall PTR.

Wang, H., & Tang, P. (2008). Preference-aware Web Service Composition by Reinforcement Learning. *In 20th IEEE International Conference on Tools with Artificial Intelligence* , 379-386.

Watkins, C. (1989). *Learning from delayed rewards.* PhD thesis, University of Cambridge.

Weld, D. (1999). Recent Advances in AI Planning. *AI MAGAZINE , 20*, 93--123.

Wellman, M., Breese, J., & Goldman, R. (1992). From knowledge bases to decision models. *Knowledge Engineering Review , 7*, 35–53.

Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., & Sheng, Q. (2003). Quality Driven Web Services Composition. *In: ACM. Proceedings of the 12th International Conference on World Wide Web (WWW)* , 411-421.