Arquitectura de aprendizaje para el manejo de riesgo de falla en ambientes de composición de servicios Web

Learning architecture for failure risk management in environments of Web services composition

Byron Enrique Portilla Rosero,[1] Esp., & Jaime Alberto Guzmán Luna, [2] MSc.
1. Estudiante de Maestría en Ingeniería de Sistemas, Universidad Nacional de Colombia - Sede Medellín
2. Profesor Asociado, Escuela de Sistemas, Universidad Nacional de Colombia - Sede Medellín
SINTELWEB: Grupo de Investigación "Sistemas Inteligentes Web"
{beportillar, jaguzman}@unal.edu.co

Recibido para revisión 14 de Abril de 2008, aceptado 25 de Agosto de 2009, versión final 07 de Septiembre de 2009

Resumen— En este artículo se presenta una arquitectura de aprendizaje basada en el aprendizaje de máquinas inductivas, la cual hace posible obtener datos durante el proceso de ejecución de servicios Web. Estos datos son analizados y con base a ellos, se genera una serie de información que se interpreta en riesgos de falla, ocasionados por los servicios Web en cualquier parte de su ejecución. Esta información es aprendida por el sistema, evitando la generación de errores en la toma de decisiones para la composición de servicios y por lo tanto, realizar una composición de servicios más precisa, utilizando aquellos servicios Web que cumplan los requerimientos solicitados sin conflictos o complicaciones durante su ejecución.

Palabras Clave—Servicios Web, Composición de Servicios Web, Manejo de Riesgos, Aprendizaje de Máquinas Inductivas.

Abstract— This paper display learning architecture based on the inductive learning machine, which allows collecting data during Web services execution. These data are analyzed, and then, a series of information is generated, which is interpreted like risks-of-failure, caused by the Web services anywhere of its execution. This information is learned by the system, having avoided the generation of errors in the decision making for the service composition and therefore, to realize a more precise service composition, using those Web services that fulfill the requirements without conflicts or complications during their execution.

Keywords— Web Services, Web Services Composition, Risk Management, Inductive Learning Machines.

I. INTRODUCCIÓN

La composición de servicios hace posible la integración de una serie de servicios Web, con el fin de satisfacer un conjunto de requerimientos solicitados, cuando un servicio Web simple no puede alcanzar dichos requerimientos [4]. Utilizando técnicas de planificación en inteligencia artificial, interactuando con heurísticas y métodos progresivos de expansión, toma aquellos servicios que cumplan ciertas condiciones con el fin de alcanzar un resultado [9]. Sin embargo, decir que un servicio Web siempre permanecerá constante y que se mantendrá en el tiempo sin sufrir alguna alteración se torna inadmisible para el mundo de la Web, debido a que ésta es un entorno dinámico, sujeto a cambios constantes ya sean ocasionados por cambios en los servicios o malas codificaciones, disponibilidad y/o mantenimiento de los mismos o problemas que rodean la Web, tales como: conexiones erróneas, caída de red, concurrencia, problemas de seguridad, codificaciones vulneradas, errores en la transmisión de datos, datos insuficientes, interpretaciones semánticas; los cuales, son reconocidos en el ambiente científico como fallos. En este punto, se habla de contextualizar aquellos factores que afecten directa o indirectamente los procesos que durante la ejecución de los servicios Web, se ven comprometidos en la toma de una decisión inestable, dentro del contexto de composición.

Uno de los inconvenientes al no elegir un servicio apropiado para cumplir un requerimiento, ocasionaría un deterioro en el tiempo que un servicio Web toma para ser ejecutado; asimismo, se podría elegir servicios Web que no son los más adecuados para ser integrados, debido a que presentan problemas como los mencionados anteriormente y, finalmente generar resultados que no serian tan confiables debido a la presencia de datos incorrectos.

Por ello, en este documento se presenta una arquitectura de aprendizaje que obtenga aquellos datos críticos durante la ejecución de los servicios, y a partir de ellos, generar un conjunto de información que valide los riesgos ocasionados y ejerza un control previo para garantizar una composición más precisa.

Este artículo explica el funcionamiento de la arquitectura de aprendizaje que apoyará la toma de decisiones, al momento de ejecutar y seleccionar un servicio acorde con los requerimientos requeridos por otros servicios o por usuarios generadores de las peticiones.

El documento está conformado por: la sección 2, presenta un enfoque general sobre la arquitectura de aprendizaje; en la sección 3, se describen los componentes utilizados por la arquitectura de aprendizaje; la sección 4, presenta los trabajos relacionados a la propuesta y, la sección 5 presenta las conclusiones y trabajos futuros.

II. ENFOQUE GENERAL DE LA ARQUITECTURA

El uso de técnicas de aprendizaje en procesos de apoyo en la toma de decisiones, ha permitido a múltiples sistemas mejorar su capacidad de análisis y presentación de resultados en entornos poco observables [10]. Dentro del contexto de la composición de servicios, existe la posibilidad de explorar y analizar datos que no se encuentran explícitos dentro del servicio Web pero que a través del estudio de algunas características y sus comportamientos, se puede llegar a la contextualización de información que permite inferir sobre los problemas que son generados dentro de este contexto y, poderlos expresar a fin de obtener mejores composiciones.

La comunidad científica ha identificado una serie de datos que describen el rendimiento y funcionamiento de los servicios. Esta, se ha concentrado enfáticamente en la utilización de parámetros de calidad QoS sobre los cuales se aplica una serie de argumentos que evalúan distintas características para cada servicio, determinando entre otros: tiempos, costos, recursos utilizados, de tal forma que pueden evaluar el comportamiento de los mismos [6][11]. En algunas ocasiones, son utilizadas varias combinaciones de las QoS que se definen como métricas; y sobre las cuales, se estima un factor de error estimado para la evaluación de eficiencia para cada uno de los servicios Web. [1][2][7]

Con base en lo anterior, esta propuesta plantea una arquitectura de aprendizaje que apoye la composición de servicios Web a través del control de riesgos que los servicios pueden generar durante el proceso de ejecución de los mismos, teniendo en cuenta sus parámetros de calidad y algunas características del entorno Web. Para ello, la arquitectura plantea un conjunto de elementos que soportan los datos generados durante la ejecución de los servicios Web, así como la proliferación de estos, en el contexto de identificación, validación, formulación y almacenamiento de los mismos y a partir de estos procesos, obtener los datos requeridos que caracterizan a cada uno de los servicios Web evaluados. Es decir, se obtiene aquellos datos considerados como relevantes en la construcción de información que permita identificar los valores de error producidos por los servicios Web y/o su entorno.

En la figura 1, se aprecia los elementos utilizados por la arquitectura de aprendizaje. Aquí se observa los elementos de identificación o ingreso de datos, el proceso de de validación de los mismos, el módulo de formulación-aprendizaje y el almacenamiento.

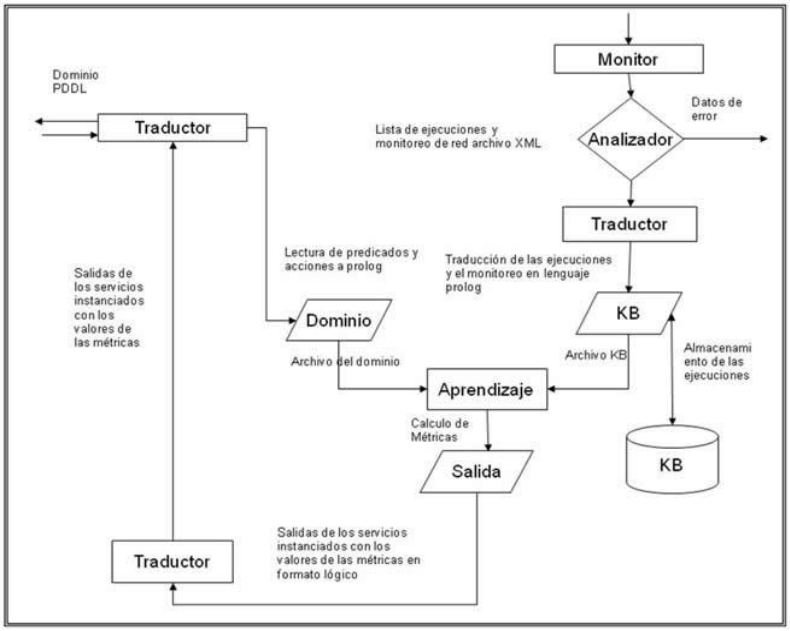

Figura 1. Arquitectura de Aprendizaje

Con esta arquitectura, se pretende aprender a partir de una serie de ejecuciones, el comportamiento del servicio Web teniendo en cuenta el monitoreo de sus salidas. Es decir, reemplazar aquellos datos que han sido establecidos por los creadores de los servicios y/o insertados por expertos donde se presume un entorno Web estático, o se simulan datos que no garantizan el total comportamiento de los servicios. Para ello, se integran los conocimientos de los QoS soportados por la composición de servicios [6][11] y sobre los cuales actúa la arquitectura de aprendizaje, evaluando principalmente aquella información que haga posible determinar que tan robusto es un servicio a través de su relación en la Web y su interacción con otros servicios.

La arquitectura cuenta con un módulo de aprendizaje, el cual se apoya en el aprendizaje inductivo, específicamente en árboles de decisión, los cuales permiten a partir de una serie de ejemplos y utilizando decisiones secuenciales basada en el uso de resultados y probabilidades asociadas, guiar los datos que evalúan a cada servicio Web. El algoritmo utilizado para tal fin es C4.5; este algoritmo, es una extensión mejorada del ID3, el cual genera un árbol de decisión a partir de los datos mediante particiones realizadas recursivamente utilizando la estrategia (depth-firts). [10]

III. DESCRIPCIÓN DE LA ARQUITECTURA DE APRENDIZAJE

La arquitectura de aprendizaje consta de un conjunto de elementos que interactúan entre sí, a partir de la obtención de información proveniente del mundo, y de información sobre el dominio en el que se está trabajando.

A. Adquisición de la Base de Conocimientos

En primer lugar, está el proceso de obtención de la información a través del ejecutor de servicios. Generalmente, la información que se obtiene del ejecutor es valor positivo o negativo dependiendo de la evaluación que realiza el ejecutor. Sin embargo, se aclara que durante este proceso, se pueden presentar varias inconsistencias que estarían relacionadas con la generación de esos valores. Es decir, no hay diferencia al momento de obtener la respuesta de fracaso generada por el ejecutor, debido a que solo está validando una comparación del servicio con el mundo, sin tener en cuenta las fallas ocasionadas por motivos exclusivos de la Web o del mismo servicio. Por ejemplo, pérdidas de transferencia de datos ocasionadas por las conexiones o privilegios de la red o los servicios que serán accedidos. Por este motivo, se suma a la caracterización del servicio un valor determinante en la decisión de la elección de un servicio, ya que si un servicio falla por motivos ajenos a este, existe una probabilidad equilibrada que ese servicio pueda ser o no ejecutado correctamente en condiciones normales.

Por otro lado, el acceso a la información que soporta cada servicio tiene un conjunto de privilegios sobre los cuales actúa para comunicar la información. Igualmente el servicio puede contener fallas internas que impiden la buena ejecución del mismo y de igual forma que la descripción de la Web, el servicio puede contener información relevante pero a la que no se puede acceder.

Con el fin de instanciar esos datos relevantes en la composición de servicios, la arquitectura incorpora un monitor localizado dentro del ejecutor de servicios, que permita hacer un seguimiento paso a paso a la ejecución de los servicios y donde haga posible analizar las fallas que se producen en un determinado proceso en el tiempo. Por lo tanto, el monitor es el encargado de abstraer aquella información considerada como relevante para el desarrollo del proceso de aprendizaje.

El monitor tiene la capacidad de hacer el seguimiento de los datos evaluados por el ejecutor y al mismo tiempo, revisa el estado de la red y sus componentes. La evaluación la realiza a través de protocolos TCP/IP.

La información resultante de este proceso es un documento XML, el cual es enviado a un traductor para ser evaluado por la arquitectura de aprendizaje.

Sin embargo, antes de esta etapa, un analizador es el encargado de verificar la consistencia de este documento y determinar si su contenido está acorde a las ejecuciones de los servicios.

Una vez que los datos son capturados, pasan por un módulo de traducción. Este elemento es el encargado de leer la codificación del monitor y trasladarla a un lenguaje de codificación lógica. El traductor lee un archivo XML de donde obtiene las sentencias de respuesta de cada servicio así como el estado de la red. Esta información es codificada al lenguaje prolog, generando un archivo de lectura rápida que representa la base de conocimientos de la arquitectura de aprendizaje.

Un fragmento de esta codificación se muestra en la figura 2. El ejemplo está dado sobre el servicio, hacer una oferta perteneciente al dominio de compras virtuales (SHOPPING).

```
currentOfferService(example1,200000,success).
atVendorItemCurrentPrice(200000,example1).
atVendorItemCurrentPrice(example1,example1).
VendorItematItem(example1,motorolaV3).

currentOfferService(example2,300000,netfail).
atVendorItemCurrentPrice(300000,example2).

currentOfferService(example3,300000,failure).
atVendorItemCurrentPrice(300000,example3)
atVendorItemCurrentPrice(example2,example3)
VendorItematItem(example3,motorolaV6)

currentOfferService(example4,400000,servfail).
atVendorItemCurrentPrice(example4,example4).
VendorItematItem(example4,motorolaU6).
```

Figura 2. Representación de la Base de conocimiento

B. Adquisición de la Información del Dominio

Con el fin de obtener la información del dominio requerido para inducir el aprendizaje, la arquitectura requiere de un archivo donde se describe el dominio que es asumido por los servicios a utilizar. Este archivo, es un documento pddl por lo tanto, se hace necesaria la utilización de un segundo traductor. En este caso el traductor codifica las expresiones descritas en pddl en instancias de inducción a un lenguaje prolog, generando un nuevo archivo que representa las especificaciones de inducción utilizadas por la herramienta de aprendizaje para la construcción de los árboles de decisión, como se muestra en las figuras 3 y 4. La figura 3 representa la descripción de los predicados en pddl y, la figura 4 representa la interpretación de los datos en el lenguaje lógico. La parte 1 representa la descripción del servicio en pddl sobre la cual, se traza el objeto de predicción y la parte 2, representa la lectura de los predicados tomados como variables de inducción.

```
(CurrentPrice_atVendorItemCurrentPrice ?CurrentPrice_domain_parameter - CurrentPrice
?_atVendorItemCurrentPrice_range_parameter - _VendorItemCurrentPrice)

(VendorItemCurrentPrice_atVendor ?_atVendorItemCurrentPrice_domain_parameter -
_VendorItemCurrentPrice ?Vendor_range_parameter - Vendor)

(VendorItemCurrentPrice_atItem ?_atVendorItemCurrentPrice_domain_parameter -
_VendorItemCurrentPrice ?Item_range_parameter - Item)
```

Figura 3 Sentencia de los predicados en pddl

```
1. // Objeto de inducción
(currentOfferService(example,CurrentPrice,class)).
classes([success,failure,netfail,servfail]).

2. // Variable de inducción
(CurrentPrice_atVendorItemCurrentPrice(CurrentPrice,example)
```

Figura 4. Fragmento del la estructura de inducción

C. Aprendizaje

La parte más importante de la arquitectura se encuentra dentro de este módulo. Es aquí donde los datos capturados en los procesos anteriormente descritos, serán evaluados para obtener la información necesaria que permita determinar el factor de riesgo de falla de un servicio Web, en un determinado espacio en el tiempo.

La evaluación consiste en determinar patrones de ejecución de los servicios. Para esto se ha definido una clasificación que relaciona la ejecución de estos, con cuatro clases: éxito, fracaso, fracaso de conexión y error del servicio (success, failure, netfail y servfail). Esta clasificación indica cuales son los posibles valores que se pueden dar en la ejecución del servicio y con las cuales, se trata de determinar la robustez del mismo.

Este módulo recibe dos archivos: la información del dominio y la información de la base de conocimientos. En el primer archivo se encuentra la definición de las acciones, que determina cual será el objetivo a predecir; igualmente, se encuentra la definición de los predicados; los cuales, determinan los patrones de inducción. En el segundo archivo se encuentra los datos generados durante la ejecución del servicio. Esta es la base de conocimiento sobre la cual se infieren los patrones que tuvo ese servicio al ser ejecutado y así determinar su comportamiento.

La representación del árbol de decisión se representa en la figura 5.

```
// Servicio ejecutado ()
Salida del servicio () ?
+--yes: [success] 0.0 [[success:0.0,failure:0.0,netfail:0.0,servfail:0.0]]
+--no:  salida del servicio () ?
      +--yes: [success] 29.0 [[success:29.0,failure:0.0,netfail:0.0,servfail:0.0]]
      +--no:  [failure] 2.0 [[success:0.0,failure:2.0,netfail:0.0,servfail:0.0]]
```

Figura 5. Árbol de decisión

Los nodos internos de la rama contienen la serie de condiciones bajo las cuales el patrón de ejecución es verdadero, mientras que los nodos de las hojas contienen las clases correspondientes.

- **Generación de la Métrica**

Dado que un servicio puede alcanzar o no una respuesta, esta, está condicionada a factores externos o internos que impiden una buena apreciación de la salida generada. Cuando un servicio se ejecuta, sus salidas son medidas y discriminadas dependiendo de su ejecución. El árbol de decisión permite obtener aquellas salidas, las cuales se encuentran clasificadas en base al instante que fueron llamadas por el ejecutor. Como se describe anteriormente, las clases son: success, failure, netfail y servfail. Por lo tanto, cada salida está condicionada a una penalización, dependiendo de su clasificación. Es decir, aquellas salidas donde su ejecución fue exitosa, no tendrán penalización. Sin embargo para los otros casos, su valor está determinado por las ecuaciones 1,2 y 3:

$$penality=1-(Ej/Ejt) \quad (1)$$

Donde Ej representa la ejecución de esa salida y Ejt representa el total de las ejecuciones.

Esta ecuación es aplicada para aquellas salidas donde no se alcanzó una respuesta, es decir son cubiertas por failure.

$$Penality=1-((Ej/Ejt)*0.5) \quad (2)$$

Cuando existen problemas que infieren en la respuesta como lo es la caída del servicio o fallos en la red, existe una probabilidad equilibrada que el servicio se ejecute satisfactoriamente o no. Por lo tanto, la penalización adiciona un valor probabilístico del cincuenta por ciento sobre el número de ejecuciones parciales y totales, manteniendo la posición de duda positiva o negativa en la ejecución.

$$Penality = Error \quad (3)$$

Un servicio puede ser calificado como *error,* cuando su estructura interna presenta problemas de configuración; por lo tanto su valor métrico tomaría un valor amplio que descalificaría ese servicio. 9999.

Después de esta evaluación, al haber identificado el comportamiento de cada servicio al ser ejecutado, se genera un último archivo con los valores métricos que serán asignados al modelo del dominio ingresado en la fase de adquisición del dominio.

D. Salida del Modelo.

La información generada durante el aprendizaje, hace posible medir la calidad de servicio, identificando los costos asignados a sus salidas.

Por lo tanto, se requiere de un último traductor que interprete la información obtenida durante el proceso de aprendizaje, y la traslade al documento del dominio; este dominio será utilizado por un planificador en la generación de nuevas composiciones.

IV. TRABAJOS RELACIONADOS

Son varios los trabajos que se han realizado en busca de mejorar la calidad de la composición de los servicios, a través del censo de QoS e igualmente, el uso de técnicas de aprendizaje durante el proceso de composición de servicios. Entre los más destacados se encuentran:

En [7] se enfocan en el manejo de riesgos en la composición de servicios. Estos riesgos hacen referencia a la forma como se manipulan las operaciones no funcionales, y los valores que tienen; lo que busca es garantizar al usuario confiabilidad en los resultados. Para ello utilizan cálculos y probabilidades, donde puedan determinar amenazas, vulnerabilidades, probabilidades e impactos que pueden mejorar o desestabilizar la composición. Los procesos que utilizan para localizar los posibles riesgos son: identificación, análisis y control de riesgo. Al tener ese monitoreo se establece si se asume o no el riesgo, debido a que existe la probabilidad de ser utilizados en otros ambientes para la solución de otros problemas. Si los riesgos son descartados se remueve el servicio.

En [11], se presenta una plataforma de software que se ocupa de la selección de los servicios web, con el propósito de su composición de manera que maximice la satisfacción de los usuarios expresada como funciones de utilidad sobre atributos de QoS, satisfaciendo las restricciones establecidas por el usuario y por la estructura de la composición de servicios. Esto se aplica tanto a los servicios Web autónomos, y servicios Web compuesto por otros servicios de la web (Servicios compuestos). Consideran la tarea de selección del servicio, como un problema global de optimización. La programación se aplica para encontrar soluciones que representen la composición del servicio que optimice la función objetivo. Al final, se define como una combinación lineal de cinco parámetros: disponibilidad, frecuencia exitosa de ejecución, tiempo de respuesta, costo de ejecución y reputación.

[3] propone un mecanismo para la propagación y recuperación de fallas en la orquestación descentralizada de servicios. La arquitectura descentralizada da lugar a la complejidad adicional, que requiere la propagación de fallas entre las particiones ejecutadas independientemente.

El problema de estas propuestas es que utilizan una técnica basada en la optimización de una función objetivo, mediante la simple suma de los pesos de los parámetros de QoS de los posibles servicios a intervenir, teniendo en cuenta su comportamiento individual (información provista por los proveedores de servicios en la mayoría de los casos), en vez de considerar su comportamiento en el propio servicio compuesto final como un todo. Lo anterior permitiría determinar de una manera más real, el factor de riesgo que presentan los servicios

elegidos en el proceso de composición, con el fin de minimizar el riesgo de falla del propio servicio compuesto resultante.

[8] busca a través de la integración de técnicas de aprendizaje y procesos de planificación y ejecución, desarrollar un planificador que pueda automáticamente mejorar el conocimiento inicial del entorno, teniendo en cuenta la ejecución de las acciones para lograr planes robustos.

V. CONCLUSIONES

En este trabajo se presentó una arquitectura de aprendizaje basada en arboles de decisión, la cual permite obtener la información crítica durante la ejecución de los servicios Web, con el fin de aprender su desempeño y optimizar el proceso de composición de servicios, a través de la generación de métricas que evalúan el comportamiento de cada servicio determinando su funcionalidad al momento de su ejecución.

Asimismo, se hizo una descripción de los componentes de la arquitectura haciendo énfasis en la obtención de la base de conocimientos y la descripción del aprendizaje orientada a la utilización de penalidades sobre la ejecución de los servicios.

Actualmente se realiza la versión inicial de la arquitectura bajo la plataforma Linux la cual, será manipulada con la ayuda de la herramienta de planificación INDIGO [5] utilizando inicialmente el dominio de compras virtuales (SHOPPING) creado dentro del mismo proyecto.

AGRADECIMIENTOS

El presente trabajo está apoyado parcialmente por el proyecto de investigación de tesis de maestría de la Escuela de Sistemas de la Universidad Nacional de Medellín "Modelo Basado en Aprendizaje de Máquinas para el Manejo de Riesgo de Falla Durante la Composición de Servicios Web" noviembre 2008.

REFERENCIAS

[1] Barreiro D.; Albers P. y Jin-Kao H., 2005. Web services composition. En: Proceedings of the ICWS 2005 Second International Workshop on Semantic and Dynamic Web Processes, Vol 3, pp. 195-225.

[2] Cardoso J.; Sheth A.; Miller J.; Arnold J. y Kochut K., 2004. Quality of service for workflows and web service processes. En: Journal of Web Semantics, Vol. 1, pp. 281-308.

[3] Chafl G.; Chandra S.; Kankar P. y Mann V., 2005. Handling faults in decentralized orchestration of composite Web services. En: International Conference on Service-Oriented Computing- ICSOC 2005, Vol. 3826, pp 410-423.

[4] Chakraborty D.; Perich F.; Joshi A.; Finin T. y Yesha Y., 2002. A reactive service composition architecture for pervasive computing environments. En: 7th Personal Wireless Communications Conference, Vol. 234, pp. 53-62.

[5] Guzmán J. y Ovalle D., 2007. Un modelo de planificación incremental para servicios Web semánticos. En: Revista Avances en Sistemas Informáticos, Vol. 4, No. 3, pp. 131-140.

[6] Jurisica I. y Nixon B., 1998. Building quality into cases based reasoning systems. En: Lecture Notes in Computer Science, Conference on Advance Information Systems Engineering, CAiSE'98 Lecture Notes in Computer Science, Vol. 1413, pp. 363-380.

[7] Kokash N., 2006. A service selection model to improve composition reliability. En: International Workshop on AI for Service Composition in conjunction with the European Conference on Artificial Intelligence (ECAI), pp. 9-14.

[8] Lanchas J.; Jiménez S.; Fernández F. y Borrajo D., 2007. Learning action durations from executions. En: Working notes of the ICAPS'07 Workshop on AI Planning and Learning.

[9] Nau D.; Ghallab M. y Traverso P., 2004 Automated Planning: Theory and Practice. Morgan Kaufman. 663 P.

[10] Quinlan J., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers. 302 P.

[11] Zeng L.; Benatallah B.; Dumas M.; Kalagnanam J. y Quan Z., 2003. Quality driven web services composition. En: Proceedings of the 12th international conference on World Wide Web, pp. 411-421.