Preference-Based AI Planning for Web Service Composition

Sebastián Vallejos, Leonardo Da Rocha Araujo, Guillermo Rodríguez, Luis Berdun and Renzo Toscani

Abstract— A web service is a technology that allows data to be exchanged between applications using open protocols and standards. Web services generally offer basic functions that solve specific problems. The composition of services occurs when more complex problems need to be solved using successive invocations to other services. The automatic composition of web services is a current problem. Due to the continuous creation and updating of services, the manual composition of new services becomes impractical. Planning algorithms are an alternative to automate the process of web services composition. In particular, the planning algorithms with preferences allow users to guide the algorithm to find a solution in line with their requirements (e.g. prioritizing the use of services according to properties such as availability, cost, response time, etc.). This paper introduces a Planning approach with preferences to compose and run complex services from a set of web services given as input. To evaluate the approach, different preferences about the input services were used to obtain composition of services. As a result, we measured the feasibility of the proposed approach and the impact of the use of preferences on the composition result.

***Index Terms*— Partial Order Planning, Web Service Composition, Planning with Preferences, Web Service Description Language, Ontology Web Service Language (OWL-S).**

I. INTRODUCCIÓN

Un servicio web es una tecnología que permite intercambiar datos entre aplicaciones mediante el uso de protocolos y estándares abiertos. Los servicios web en general ofrecen funciones básicas o simples que solucionan problemas puntuales. Estos servicios no siempre pueden cumplir con los requisitos del cliente por sí mismos, por lo que en esos casos puede elegir hacer una composición de los servicios web [1].

S. Vallejos, Universidad Nacional del Centro de la Provincia de Buenos Aires, Tandil, Buenos Aires, Argentina, (e-mail: sebastian.vallejos@isistan.unicen.edu.ar).
L. Da Rocha Araujo, Universidad Nacional del Centro de la Provincia de Buenos Aires, Tandil, Buenos Aires, Argentina, (e-mail: leonardo.araujo@isistan.unicen.edu.ar).
G. Rodríguez, Universidad Nacional del Centro de la Provincia de Buenos Aires, Tandil, Buenos Aires, Argentina, (e-mail: guillermo.rodriguez@isistan.unicen.edu.ar).
L. Berdun, Universidad Nacional del Centro de la Provincia de Buenos Aires, Tandil, Buenos Aires, Argentina, (e-mail: luis.berdun@isistan.unicen.edu.ar).
R. Toscani, Universidad Nacional del Centro de la Provincia de Buenos Aires, Tandil, Buenos Aires, Argentina, (e-mail: renzo.toscani@gmail.com).

La composición de servicios se da cuando se necesita solucionar problemas mayores utilizando sucesivas invocaciones a distintos servicios. Esta composición puede realizarse de forma manual, pero la continua creación y actualización de los servicios web, hace que esto sea impracticable [2]. Por esta razón, la composición automática de servicios web es una problemática actual.

Distintas alternativas surgieron para la composición de servicios web de manera automática [3], entre ellas, el uso de algoritmos de planeación (planning) [4]. Planeación es una rama de la Inteligencia Artificial (IA) que tiene como objeto la creación de secuencias de acciones que dado un estado inicial conocido permiten alcanzar un estado final deseado [5]. Los algoritmos de planeación se encargan de elegir y ordenar las acciones a realizar para lograr determinado objetivo a partir de cierto estado inicial. Estos algoritmos son en su mayoría de propósito general y pueden aplicarse en infinidad de dominios [3, 6, 7].

En nuestro caso particular, nos permiten componer distintos servicios para poder alcanzar el servicio buscado. Algunos algoritmos de planeación permiten adicionalmente indicar preferencias o distintos criterios extras a la hora de realizar el plan de ejecución. El algoritmo AG UCPOP fue diseñado para interactuar con el estado mental de agentes. Por esta razón permite definir actitudes mentales (preferencias y objetivos) mediante lógica proposicional. Estas actitudes impactan en la forma en que el algoritmo alcanza la solución final [8]. AG UCPOP mantiene su premisa básica, es decir que permite seleccionar y ordenar las acciones para alcanzar el estado final planteado, pero lo hace buscando soluciones que no solo resuelven el problema, sino también que se adecúen a lo pretendido.

Como es posible encontrar diferentes formas en que se compongan los servicios para que cumplan con los requisitos del usuario, es necesario algún mecanismo para escoger cuál es la mejor combinación de servicios. En este punto los algoritmos de planeación con preferencias hacen la diferencia, permitiendo al usuario guiar al algoritmo para encontrar una solución más acorde a sus requisitos (por ejemplo, priorizar el uso de servicios según propiedades como disponibilidad, costo, tiempo de respuesta, etc). Por lo tanto, dicho algoritmo seleccionará de un conjunto de servicios, el que mejor cumpla con las preferencias del usuario.

Este trabajo presenta un enfoque para componer y ejecutar servicios complejos a partir de un conjunto determinado de servicios web más simples. Para tal fin se utilizarán algoritmos de planeación que aceptan preferencias del usuario para obtener no solo el servicio sino también la calidad buscada por el usuario. Durante las evaluaciones se utilizarán distintas preferencias sobre los servicios para lograr obtener la composición de servicios deseada. De esta forma se logra medir la factibilidad del enfoque propuesto y el impacto del uso de preferencias en el resultado de la composición.

El resto del trabajo se organiza de la siguiente manera. La

sección 2 describe los trabajos relacionados. La sección 3 presenta el enfoque propuesto. La sección 4 reporta los resultados experimentales. La sección 5 discute los resultados obtenidos. Finalmente, la sección 6 concluye nuestro trabajo e identifica futuras líneas de trabajo.

II. Trabajos Relacionados

La computación evolutiva y heurísticas de optimización multi-objetivo han sido muy usadas para la composición automática de servicios web [3]. Por ejemplo, los autores en lugar de considerar los servicios individualmente durante la composición, proponen un conjunto de abstracciones y refinamientos para formar grupos de servicios basados en características funcionales. Este método ofrece una aceleración significativa en comparación con las técnicas de composición tradicionales, ya que solamente se explora un espacio sustancialmente más pequeño [9]. En esta línea, los autores en [10] proponen un modelo de optimización multi-objetivo (EDMOEA) en el que el rendimiento de QoS (calidad de servicio) y la variación de las restricciones de QoS del usuario son objetivos independientes. El método propuesto intenta encontrar el óptimo de Pareto que ayuda a los usuarios a seleccionar los servicios adecuados con la compensación del rendimiento y el riesgo de QoS. En [11], los autores plantean la composición de servicios web considerando QoS desde una perspectiva de problema de optimización multi-objetivo. Los autores han comparado varios algoritmos como NSGA-II, SPEA2, MOEA/D, GDE3 y POSDE; y concluyeron que GDE3 es la mejor opción para el problema de composición de servicios considerando QoS.

En [12], se propone el uso de la programación genética para la composición de servicios Web. Como resultado, se muestra que es probable que exista una compensación entre el tiempo de ejecución y la calidad de las soluciones cuando se emplea programación genética y *Particle Swarm Optimization*. Esto lleva a la conclusión de que los enfoques de programación genética pueden ofrecer una solución escalable al problema de la composición automatizada de servicios web.

En cuanto a los algoritmos de planeación, estudios recientes han mostrado avances significativos en el uso de planeación para la composición de servicios web [13, 14]. En [13], los autores han propuesto *Q-GraphPlan* basado en *GraphPlan* clásico, un planificador eficiente para resolver problemas de planeación clásicos. Primero, construyen un grafo de planeación basado en las relaciones de dependencia de los servicios web y se extraen heurísticas esenciales de acuerdo con el análisis de accesibilidad. En segundo lugar, se convierte este grafo en un gráfico de generación de ruta dirigida. Finalmente, extraemos la solución óptima del gráfico de generación dirigido utilizando un algoritmo A* *backwards* con la heurística del grafo de planeación. En [14], los autores presentaron un enfoque de composición de servicios web dinámico basado en un algoritmo de planeación *Blackbox*. Los resultados experimentales han mostrado la efectividad del enfoque propuesto.

El trabajo presentado en [4], los autores desarrollaron un algoritmo capaz de traducir descripciones de servicios del lenguaje OWL (Ontology Web Language) en objetos de dominio SHOP2 para, posteriormente, realizar el planeamiento y la ejecución de los servicios web. Sin embargo, este trabajo no considera las preferencias del usuario a la hora de componer los servicios. Siguiendo la línea de planeación jerárquica, el enfoque abordado en [15] incluye las preferencias del usuario a la composición automática de servicios. Los algoritmos HTN son útiles cuando se conoce formas de descomponer acciones complejas en más simples. Es decir que se necesita conocer previamente el dominio y como se puede componer el mismo para aprovechar el mismo. Este punto es importante de destacar ya que, en nuestro caso, la composición se aprende durante la ejecución del algoritmo. Asimismo, al usar las preferencias definidas por algoritmos (PDDL3), éstas se evalúan si se cumplen o no sobre el plan alcanzado (preferencia sobre estados del plan). Esto es importante ya que no se contempla la cuantificación de las preferencias. Esta diferencia es fundamental con nuestro enfoque, el cual maximiza la utilización de las preferencias y, además, contempla la ejecución de los servicios.

Otros trabajos que han explorado la incorporación de preferencias de usuarios para composición de servicios se describen a continuación. En [16], los autores han propuesto un enfoque para permitir a los usuarios componer servicios personalizados sin necesidad de especificación manual; para ellos, han propuesto un algoritmo de *learning-to-rank* para aprender automáticamente las preferencias del usuario y la priorización de las preferencias a partir de los datos históricos de los usuarios. En esta línea, en [17], los autores han propuesto un modelo de composición que considera propiedades no funcionales tanto cuantitativas como cualitativas. Los autores han desarrollado 2 algoritmos: el primero combina la optimización global con la selección local, mientras que el segundo usa un algoritmo genético. Para ello, se usan *Tradeoffs-enhanced CP-nets* (*Conditional Preference Networks*) para modelar las preferencias del usuario.

Las investigaciones realizadas hasta el momento proporcionan un antecedente para la composición de servicios, sin embargo, no existen trabajos que permitan realizar la ejecución de los servicios en conjunto con la introducción de preferencias por parte de los usuarios. Adicionalmente, las herramientas existentes requieren que los usuarios conozcan tanto aspectos técnicos como de software e IA para la composición dinámica de los servicios utilizando preferencias.

Este trabajo propone un enfoque que aborda esta cuestión mediante el algoritmo de planeación AG UCPOP [8], el cual permite al usuario especificar, además de preferencias directas, las propiedades de los servicios sobre las cuales el usuario puede establecer su preferencia.

A. Algoritmo AG UCPOP

El algoritmo AG UCPOP consiste en una extensión del tradicional algoritmo UCPOP [18]. UCPOP es, a su vez, una extensión del algoritmo de planeación de orden parcial que genera planes mediante una técnica de búsqueda hacia atrás. Este algoritmo toma como entrada un estado inicial del mundo y una lista de acciones que permiten modificar el mundo. Las acciones especifican un conjunto de precondiciones requeridas para poder aplicarlas y las modificaciones que realizan en el mundo.

Básicamente, UCPOP funciona de la siguiente manera: dada una lista de objetivos, en cada paso el algoritmo UCPOP

selecciona un objetivo de la lista y busca una acción que cumpla con dicho objetivo (una acción nueva o una acción ya existente en la planeación). Si era una acción nueva, las precondiciones de la misma son incorporadas a la lista de objetivos. Esto se repite hasta que la lista de objetivos esté vacía. En este punto, la planeación alcanzada constituye el plan final. Si en un momento no se encuentra una acción para el objetivo elegido, se deshace el paso anterior. Durante su ejecución, el algoritmo considera la noción de menor compromiso, por la cual se incorporan variables y no se les asocia un valor hasta que sea necesario (si se las puede vincular, por ejemplo: se puede establecer que "A debe igual a B" sin que estas variables tengan todavía algún valor asociado).

En particular, el algoritmo AG UCPOP fue desarrollado con el objetivo de permitir a sistemas de agentes generar planes de ejecución que satisfagan sus actitudes mentales. Este algoritmo permite incorporar parámetros opcionales (preferencias y restricciones) que ajustan la manera en la que el algoritmo UCPOP selecciona y combina las acciones que deben ejecutarse. En cada momento que se realice una decisión el algoritmo considera el estado mental del agente. Por ejemplo, si se tienen varias acciones que cumplen el mismo objetivo, el algoritmo selecciona primero la que aporte una mayor preferencia para el agente. Las preferencias se escriben de la siguiente forma en AG UCPOP:

preference (Q, AC, AD, P, C) :- B

Donde Q representa el objetivo que se busca cumplir cuando se selecciona la acción, AC es la acción candidata a tener preferencia, AD es la acción destinataria, P es el peso de la preferencia; cuanto mayor es el número, mayor es la preferencia hacia esa acción. C representa la confianza sobre la preferencia. Finalmente, B es el cuerpo de la preferencia: indica las condiciones que se deben cumplir para que la preferencia sea válida.

Además de admitir preferencias, el algoritmo AG UCPOP permite definir 2 tipos de restricciones: restricciones sobre las acciones y restricciones sobre el plan. Las restricciones sobre las acciones permiten restringir el uso de cierta acción en el plan, mientras que las restricciones sobre el plan permite restringir ciertas combinaciones de acciones sobre el plan.

En el dominio de servicios web, cada servicio se puede representar como una acción del algoritmo de planeación: las entradas del servicio web serían las condiciones de la acción, mientras que la salida de los servicios web constituyen los efectos de la acción. Por su parte, las preferencias del usuario a ciertos atributos de calidad de los servicios (por ejemplo, servicios gratuitos o de alta disponibilidad) pueden representarse como preferencias del agente a ser consideradas durante la formulación de un plan de ejecución.

III. Enfoque Propuesto

El enfoque propuesto se presenta en la Fig. 1. Los objetivos del enfoque son:

- Interpretar y modelar los servicios de forma tal que el algoritmo de planeación logre identificar las entradas y salidas de los servicios compatibles.
- Planificar los servicios que deben ser ejecutados mediante la introducción de preferencias al algoritmo de planeación.
- Ejecutar los servicios que se encuentran en los planes generados por el algoritmo de planeación.

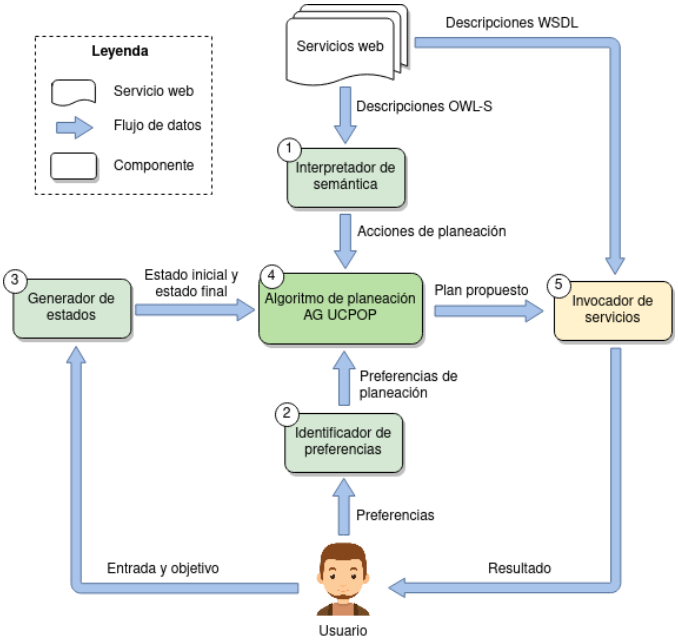


Fig. 1. Enfoque propuesto.

El enfoque comienza con la definición de los servicios web disponibles usando OWL-S (Web Ontology Language for Services). Estas descripciones ingresan al componente ① denominado *Interpretador de semántica*, que se encarga de interpretar la semántica de los servicios web para inferir cuáles son compatibles entre sí. OWL-S proporciona una descripción de los servicios a nivel de proceso que, además de información funcional, modela las condiciones previas y posteriores de los procesos para que la evolución del dominio pueda inferirse lógicamente [19].

A partir de las descripciones OWL-S, el componente ① genera acciones de planeación, que resultan de transformar cada servicio web en una acción que el componente ④, denominado *Algoritmo de Planeación,* sea capaz de interpretar. Estas acciones contienen pre-condiciones y post-condiciones que mapeadas a las entradas y salidas de cada uno de los servicios.

Luego, para poder ejecutar el algoritmo de planeación es necesario que el usuario establezca las variables de entrada del mismo. Por un lado, el Usuario debe indicar las preferencias para la selección de servicios. En este sentido, componente ②, denominado *Identificador de preferencias* es el encargado de identificar y traducir los requisitos del Usuario en preferencias de planeación. Por ejemplo, el Usuario podría indicar que prefiere utilizar servicios que no tengan costo monetario asociado. Estas preferencias permiten guiar al algoritmo de planeación hacia una solución compatible con los requisitos del Usuario.

Por otro lado, el Usuario deberá establecer los datos de entrada y el objetivo deseado, es decir, el conjunto de

variables a satisfacer mediante la ejecución de los servicios web. En este caso, el componente ③, denominado *Generador de estados* será el encargado de traducir y transformar los datos de entrada y el objetivo en el estado inicial y estado final (respectivamente), para que el algoritmo de Planeación sea capaz de interpretarlo.

En este punto, el componente ④, *Algoritmo de Planeación,* ya posee todos los datos para ejecutar: las preferencias, el estado inicial, el estado final y las acciones. Utilizando estos datos algoritmo AG UCPOP genera un plan de ejecución, denominado plan propuesto. Este plan detalla las acciones que deben ejecutarse (cada acción correspondiente a un servicio web) y las dependencias entre ellas (que acciones deben ejecutarse antes que otras).

Una vez obtenido el plan propuesto, el componente ⑤ denominado *Invocador de Servicios*, junto con las descripciones en el lenguaje de descripción de servicios web (*Web Service Description Language*, WSDL) de los servicios web disponibles, es el encargado de obtener el resultado final de la composición para el Usuario. En este sentido, se ejecutan todos los servicios web correspondientes a las acciones del plan propuesto. Luego de ejecutar exitosamente todos los servicios, se retorna el resultado al Usuario. En las siguientes sub-secciones se muestra en detalle cada componente del enfoque propuesto.

A. Interpretador de Semántica

Para que el algoritmo de planeación sea capaz de interpretar los servicios, es necesario definirlos en un lenguaje para poder comprender la semántica de las acciones y lo que conlleva cada una de ellas. Para ello, se utilizan los lenguajes *Web Service Modeling Ontology* (WSMO) [20] y *Web Ontology Language for Services* (OWL-S) [21], que extienden el poder descriptivo de *Web Service Description Language* (WSDL). Tanto WSMO como OWL-S permiten modelar estados del mundo, objetivos por los cuales un usuario debería llamar cierto servicio web, las variables afectadas por la llamada del mismo, precondiciones, y postcondiciones, entre otros elementos semánticos. Esta descripción semántica de los servicios web contiene la información que el algoritmo de planeación necesita para poder organizar las invocaciones a los servicios web.

La ventaja de OWL-S por sobre WSMO es que OWL-S permite describir procesos compuestos por otros procesos. De esta forma se puede describir los planes creados por el algoritmo de planeación como nuevos procesos compuestos, enriqueciendo con la variedad de procesos, evitando así planear dos veces para un mismo objetivo.

```
<profile:Profile rdf:ID="perfilSumar">
  <profile:serviceName>Sumar</profile:serviceName>
  <profile:textDescription>
    Este servicio permite sumar dos números
  </profile:textDescription>
  <profile:costOfService>
    <profile:nonCost/>
  </profile:costOfService>
  <profile:availability>
    70
  </profile:availability>
  <profile:has_process>
    &process;#sumar
  </profile:has_process>
</ profile:Profile>
```

Fig. 2. Perfil para el servicio *Sumar* utilizado como ejemplo conductor.

Para entender mejor la descripción de los documentos y su posterior uso es necesario definir un ejemplo conductor. En este caso particular utilizaremos un ejemplo que consiste en un servicio web denominado *Sumar*, que obtiene la suma de dos números recibidos como entrada. El perfil de este servicio (detallado en la Fig. 2) ilustra al servicio *Sumar* con su descripción, sin costo y con un nivel de *availability* de 70.

```
<process:AtomicProcess rdf:ID="sumar">

  <process:hasInput>
    <process:Input rdf:ID="Sumando1">
      <process:parameterType rdf:datatype="&xsd;#anyURI">
        &ontology;#Numero
      </process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="Sumando2">
      <process:parameterType rdf:datatype="&xsd;#anyURI">
        &ontology;#Numero
      </process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasOutput>
    <process:Output rdf:ID="Total">
      <process:parameterType rdf:datatype="&xsd;#anyURI">
        &ontology;#Numero
      </process:parameterType>
    </process:Output>
  </process:hasOutput>

  <process:hasResult>
    <process:Result>
      <process:hasEffect>
        <expr:SWRL-Expression>
          <expr:expressionObject>
            <swrl:AtomList>
              <rdf:first>
                <swrl:IndividualPropertyAtom>
                  <swrl:propertyPredicate
rdf:resource="&ontology;#suma"/>
                  <swrl:argument1 rdf:resource="#Sumando1"/>
                  <swrl:argument2 rdf:resource="#Sumando2"/>
                  <swrl:argument3 rdf:resource="#Total"/>
                </swrl:IndividualPropertyAtom>
              </rdf:first>
              <rdf:rest rdf:resource="&rdf;#nil"/>
            </swrl:AtomList>
          </expr:expressionObject>
        </expr:SWRL-Expression>
      </process:hasEffect>
    </process:Result>
  </process:hasResult>

</process:AtomicProcess>
```

Fig. 3. Proceso del servicio *Sumar* usado como ejemplo conductor.

A su vez, el perfil del servicio *Sumar* asocia al servicio con un proceso atómico denominado *sumar* (detallado en la Fig. 3). Este proceso muestra la definición de los dos sumandos de entrada al servicio (*Sumando1* y *Sumando2*), la salida denominada *Total* y una post-condición que establece la relación *suma* entre los dos *Sumandos* y el *Total*.

Comprender la semántica de los servicios web (qué entra y qué sale de cada uno de ellos) es central para llevar a cabo la composición automática. Supongamos un servicio web *CityTemp* que devuelva la temperatura de una ciudad a partir de su código postal. El algoritmo de planeación necesita conocer el significado de cada variable para componer los servicios con coherencia. Por ejemplo, durante la planeación es importante saber qué variables representan códigos postales de ciudades para saber cuáles pueden ser utilizadas de entrada en el servicio web *CityTemp* y no invocar el servicio con una variable numérica cualquiera. De la misma manera, es necesario conocer el significado semántico de las variables

retornadas por el servicio para saber si se pueden utilizar como entrada en otros servicios, y de esa manera poder llevar a cabo la composición.

Una vez que se han definido el proceso y el perfil, el siguiente paso es representar al servicio como una acción. En consecuencia, será necesario definir los siguientes elementos para realizar la traducción a la acción:

- *Argumentos:* Representa la lista de inputs de la acción.
- *Condiciones:* Permiten definir condiciones directas sobre las variables sin instanciar para que la acción sea aplicable, como equivalencias o diferencias entre ellas. Este tipo de condiciones no se utiliza en el presente trabajo dado que es generalmente inusual este tipo de restricciones en un servicio web.
- *Pre-condiciones:* Permite definir las precondiciones necesarias en el plan para poder agregar la acción al mismo.
- *Post-condiciones:* Representan los efectos producidos por la acción.

De esta manera, una acción queda definida de la siguiente forma:

action (nombre_de_accion (argumentos), [condiciones], [precondiciones], [postcondiciones]).

La Fig. 4 ilustra la acción generada para el servicio *Sumar*. En la acción, se puede observar el nombre de la acción (*sumar*), los argumentos *Sumando1* (*V0*), *Sumando2* (*V1*) y *Resultado* (*V2*). La lista condiciones se procede a dejar en vacío como se mencionó anteriormente. La lista de pre-condiciones tiene los siguientes hechos: *numero (V0)* y *numero(V1)*. La lista de post-condiciones contiene los siguientes hechos: *suma (V0, V1, V2)* y el hecho *numero(V2)* que muestra que el resultado V2 es un número.

```
action(sumar(V0,V1,V2),
  [],
  [numero(V0),numero(V1)],
  [suma(V0,V1,V2),numero(V2)]
).
```

Fig. 4. Acción generada para el servicio *Sumar*.

B. Generador de Estados e Identificador de Preferencias

Luego de traducir los servicios web a acciones de planeación, se deben definir: (i) el estado inicial y el estado final para que el algoritmo de planeación tenga un punto de partida y un objetivo; y (ii) las preferencias del Usuario, para que el algoritmo pueda ajustarse y generar un plan con los servicios que mejor se adecuen a los deseos del Usuario.

En primer lugar, se definen el estado inicial y final. El estado inicial se define como un conjunto finito de variables y predicados instanciados. En cambio, el estado final se define como un conjunto finito de variables y predicados que pueden estar sin instanciar. A la hora de ejecutar, el algoritmo de planeación deberá encontrar una secuencia de acciones que permita llegar desde el estado inicial, hasta el estado final.

Continuando con el ejemplo del servicio *Sumar*, supongamos que el usuario desea obtener el resultado de sumarle 7 a la suma entre 2 y 5. La Fig. 5 (a) muestra el estado inicial de esta situación, donde se tienen definidos los números 2, 5 y 7. En la Fig. 5 (b), el estado final define la variable objetivo V1, resultante de sumar a 7, la variable objetivo V0 que, a su vez, es el resultado de la suma entre 2 y 5.

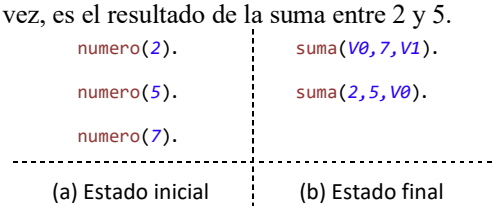


Fig. 5. Ejemplo de (a) estado inicial y (b) estado final.

En segundo lugar, se definen las preferencias del Usuario. El algoritmo AG UCPOP permite introducir preferencias del usuario en forma de expresiones lógicas. Estas expresiones definen el orden de las acciones a considerar para el plan actual en caso de que varias de estas acciones cumplan con los requisitos, es decir, las preferencias que valorizan una acción por sobre las demás.

De esta forma, si se tienen varias acciones que cumplen el mismo objetivo, el algoritmo selecciona la de mayor preferencia. Las preferencias serán establecidas automáticamente en base a los atributos definidos en los perfiles de los servicios. Esto servirá para establecer una preferencia inicial en base a los atributos de los servicios web. Algoritmo de Planeación e Invocador de Servicios

En este punto, se poseen las entradas necesarias para que el algoritmo de planeación genere un plan. El algoritmo AG UCPOP toma los datos detallados en las sub-secciones anteriores: la lista de acciones, el estado inicial, el estado final y las preferencias del Usuario. Con estos datos, se propone un plan de ejecución para lograr alcanzar el estado final a partir del estado inicial, aplicando una secuencia de acciones.

El plan propuesto resultante es un plan de orden parcial. Un plan de orden parcial se puede ver como un conjunto de acciones que deben ser ejecutadas en orden. El plan en cuestión puede tener más de una acción en el mismo nivel, esto significa que las acciones son totalmente independientes entre sí y pueden ser ejecutadas de forma paralela. Contrariamente, las acciones que están en distintos niveles son dependientes de alguna de las acciones que aparecen en los niveles inferiores.

Una vez obtenida la lista ordenada de acciones se procede a la interpretación y ejecución del plan. Cada acción del plan se reemplaza por el servicio web correspondiente. Luego, se recorre la lista de servicios ejecutándolos en orden y almacenando los resultados de cada uno para ser consumidos por otros servicios (en caso de que el plan propuesto así lo defina). Finalmente, si todos los servicios se ejecutan de manera exitosa se presenta la solución al Usuario.

IV. Resultados Experimentales

El objetivo de evaluar el enfoque propuesto es medir el impacto que generan las preferencias establecidas por un usuario para la composición y ejecución de servicios web. Para medir este impacto, se utilizarán 2 casos de estudio y 2 perfiles de usuario con diferentes preferencias.

El primer caso de estudio consta de un conjunto de servicios que permiten obtener distintos parámetros del clima de una ciudad en particular, mientras que el segundo caso de estudio consta de un conjunto de servicios que permiten obtener datos de vuelos, hospedajes, tours y transportes necesarios para realizar un viaje. El perfil del usuario U1 está definido considerando que elegir un servicio con costo monetario sería 10 veces peor que elegir un servicio sin costo monetario. En este sentido, se define con el valor -1 de preferencia si el servicio no tiene costo monetario asociado, y con valor -10 si tiene costo.

En cambio, el perfil del usuario U2 está definido contemplando que además del costo monetario, el usuario considera la disponibilidad del servicio. Entonces, un servicio sin costo monetario y con alta disponibilidad tendrá un valor de preferencia -1; un servicio con costo monetario y con baja disponibilidad tendrá un valor de preferencia -2; un servicio con costo monetario asociado y con alta disponibilidad tendrá un valor de -10; y finalmente, un servicio con costo monetario y con baja disponibilidad tendrá un valor de preferencia -20.

De esta manera, se define el impacto (1) como una función cuyo objetivo es maximizar la preferencia global de la composición de servicios resultante:

$$\sum_{i=1}^{n} Preferencia_i \qquad (1)$$

Los servicios resultantes de una composición pueden considerarse como una tupla C=<S_1, S_2,…, S_n>, con cada servicio S_i tiene una *Preferencia*$_i$ asociada según el perfil del usuario. En este contexto, de acuerdo a la representación del problema, el impacto óptimo será el más cercano a 0.

A. Setup del Experimento

El primer caso de estudio consta de un conjunto de servicios que permiten obtener diferentes datos climáticos de una ciudad específica, como también realizar la conversión entre diferentes unidades de medida.

La Tabla 1 define el conjunto de servicios con sus respectivos parámetros de entrada (entradas), sus valores de retorno (salidas), y los valores de preferencia según los perfiles de los usuarios U1 y U2. Es importante aclarar que estos valores de preferencia no se establecieron de manera particular para cada servicio web, sino que se calcularon según la categoría de cada servicio web y los perfiles de los usuarios U1 y U2 detallados previamente. En la experimentación, se considera que un servicio tiene baja disponibilidad si no supera el umbral asignado con valor 50. Si un servicio supera ese umbral, se lo considera con alta disponibilidad.

TABLA I
DESCRIPCIÓN DE LOS SERVICIOS EN EL CASO DE ESTUDIO #1.

Nombre del Servicio	Entradas	Salidas	U1	U2
Servicios de baja disponibilidad y con costo				
CityForecast	CityZip	Temperature Celsius, WindKph, RainProbability	-10	-20
Servicios de alta disponibilidad y con costo				
Pressure	CityZip	Pressure	-10	-10
Servicios de alta disponibilidad y sin costo				
Temperature Celsius	CityName	Temperature Celsius	-1	-1
Temperature Kelvin	CityName	Temperature Kelvin	-1	-1
Temperature Fahrenheit	CityName	Temperature Fahrenheit	-1	-1
WindMph	CityName	WindMph	-1	-1
Servicios de baja disponibilidad y sin costo				
WindKph	CityName	WindKph	-1	-2
RainProbability	CityName	RainProbability	-1	-2
Humidity	CityName	Humidity	-1	-2
FahrenheitTo Celsius	Temperature Fahrenheit	Temperature Celsius	-1	-2
FahrenheitTo Kelvin	Temperature Fahrenheit	Temperature Kelvin	-1	-2
KelvinTo Fahrenheit	Temperature Kelvin	Temperature Fahrenheit	-1	-2
KelvinTo Celsius	Temperature Kelvin	Temperature Celsius	-1	-2
CelsiusTo Kelvin	Temperature Celsius	Temperature Kelvin	-1	-2
CelsiusTo Fahrenheit	Temperature Celsius	Temperature Fahrenheit	-1	-2
CityNameTo CityZip	CityName	CityZip	-1	-2
CityZipTo CityName	CityZip	CityName	-1	-2
WindKphTo Mph	WindKph	WindMph	-1	-2
WindMphTo Kph	WindMph	WindKph	-1	-2

La Fig. 6 muestra nuestro enfoque en acción. Supongamos que un usuario desea consumir un servicio web que permita convertir una temperatura de la ciudad de *Tandil* (estado inicial) medida en grados Celsius a grados Kelvin. Para ello, el usuario ingresa sus preferencias: consumir servicios sin costo (*cost = false*), de alta disponibilidad (*availability = 65*) y de alta preferencia para el usuario (*preference = -1*). Como resultado, los servicios obtenidos por AG UCPOP son 2: *TemperatureCelsius* y *CelsiusToKelvin* (resaltado en Tabla 1). En primer lugar, el primer servicio retorna el valor 15°C. Este valor ingresa como parámetro de entrada al segundo servicio, el cual retorna 288.16°K. Finalmente, se obtiene la temperatura en grados Kelvin y se la asocia a la ciudad de *Tandil*, dada como parámetro de entrada (estado final).

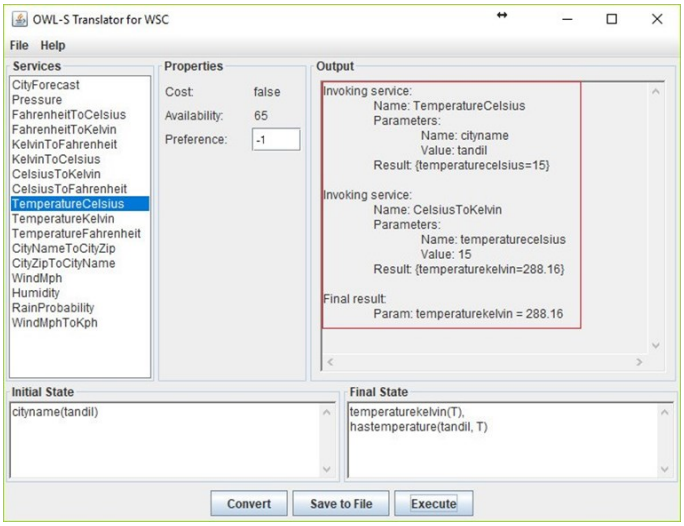


Fig. 6. Nuestro enfoque en acción.

El segundo caso de estudio se encuentra compuesto de servicios que permiten obtener información de las diferentes etapas que forman parte de un viaje. En la Tabla 2 se detalla el conjunto de servicios presente en este caso de estudio de manera similar a como se detallaron los servicios del primer caso de estudio en la Tabla 1.

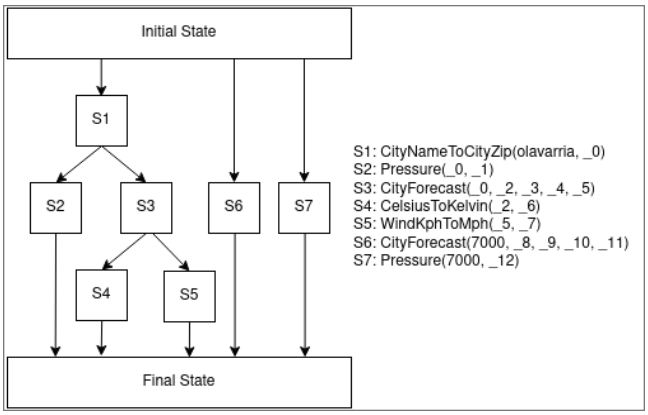
TABLA II
DESCRIPCIÓN DE LOS SERVICIOS EN EL CASO DE ESTUDIO #2.

Nombre del Servicio	Entradas	Salidas	U1	U2
Servicios de alta disponibilidad y con costo				
BookTravel	Date, Date, CityName	Hotel, Flight, Transport, Tours	-10	-10
Servicios de baja disponibilidad y sin costo				
BookHotel	Date, Date, CityZip	Hotel	-2	-2
BookFlight	Date, Date, CityZip	Flight	-2	-2
Book Transport	Date, Date, CityZip	Transport	-2	-2
BookTours	Date, Date, CityZip	Tours	-2	-2
Servicios de alta disponibilidad y son costo				
CityNameTo CityZip	CityName	CityZip	-1	-1
CityZipToCityName	CityZip	CityName	-1	-1

B. Caso de Estudio #1: Servicios de Clima

Para este caso de estudio se realizaron 2 pruebas diferentes, utilizando preferencias y sin la utilización de las mismas. El objetivo se basa en obtener todos los datos del clima para las ciudades de *Tandil* y *Olavarría*. Los datos del clima que se desean obtener son los siguientes: (i) Temperatura expresada en grados centígrados, (ii) Velocidad del viento en kilómetros por hora, (iii) Probabilidad de precipitaciones, (iv) Porcentaje de la humedad en el ambiente, (v) Presión atmosférica.

Para la primera prueba, el objetivo consistió en obtener una composición de servicios sin considerar las preferencias del usuario U1. De esta manera, la composición resultante contiene una cantidad limitada de servicios, los cuales pueden tener un costo asociado. A modo de ejemplo, la Fig. 7 muestra



el resultado final junto con los estados inicial y final.

Fig. 7. Solución para la primera prueba del primer caso de estudio.

Luego, se realizó un segundo experimento dentro de primera prueba, donde se consideró el perfil de U1. Este perfil prefiere servicios priorizar la elección de servicios sin costo asociado. De esta forma, se privilegia a los servicios gratuitos que permiten obtener los diferentes datos del clima así como las transformaciones de unidades; en segundo lugar se priorizan los servicios con costo asociado.

En la segunda prueba, el objetivo consistió en obtener una composición de servicios sin considerar las preferencias del usuario U2. En primer lugar, se obtuvo una composición sin considerar la preferencia de U2 sobre los servicios sin costo asociado. En segundo lugar, se realizó nuevamente la composición considerando el perfil de U2, el cual además de priorizar los servicios sin costo, prioriza servicios con alta disponibilidad. Por ejemplo, en la Fig. 8 los servicios *TemperatureCelsius*, *TemperatureFahrenheit*, y *WindMph* han sido seleccionados dado que no tienen costo asociado y poseen alta disponibilidad.

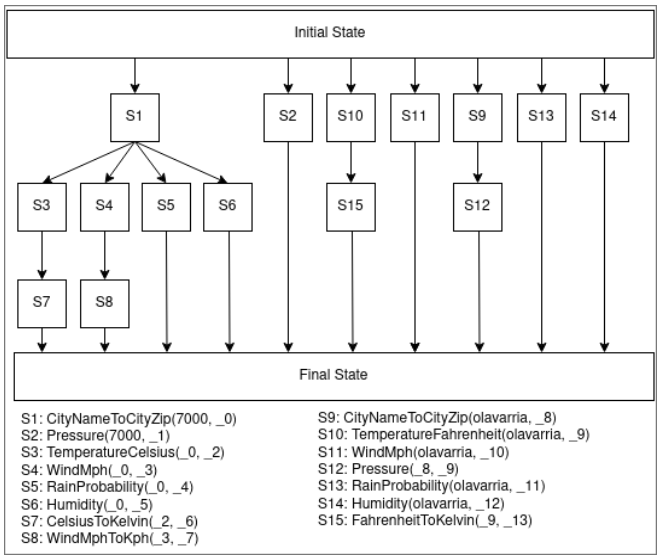


Fig. 8. Solución para la tercera prueba del primer caso de estudio.

En la Tabla 4 se puede apreciar los resultados de las pruebas y, a su vez, comparar el valor total de las preferencias cumplidas cuando se toman en cuenta las preferencias del usuario. En la tabla se puede ver resaltado que el impacto se maximiza cuando se consideraron las preferencias de los usuarios.

TABLA IV
RESULTADOS DEL CASO DE ESTUDIO #1.

	Impacto con Preferencias U1	Impacto con Preferencias U2
Composición sin preferencias	-43	-66
Composición con preferencias	**-30**	**-42**

C. Caso de Estudio #2: Servicios de Viaje

Para este caso de estudio se llevaron a cabo 2 pruebas que involucran tanto la utilización de preferencias como la ausencia de las mismas. En todos ellos se desea obtener toda la información necesaria para llevar a cabo un viaje hacia *Londres*, es decir, obtener los vuelos, hotel, transporte y excursiones en el destino.

En la primera prueba, se obtuvo una composición de servicios prescindiendo de las preferencias de U1, que prioriza

los servicios sin costo asociado. Esto dio lugar a que la solución obtenida se componga únicamente por el servicio *BookTravel*. En segundo lugar, se utilizaron las preferencias de U1 y se obtuvo el plan que se muestra en la Fig. 9. Los servicios utilizados en cuestión son: *BookHotel*, *BookTours*, *BookTransport, BookFlight* y *CityNameToCityZip*.

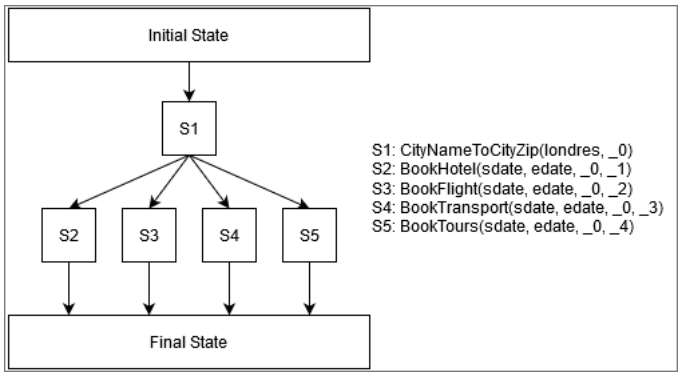


Fig. 9. Solución para la segunda y tercera prueba del segundo caso de estudio.

En la segunda prueba, se tuvo en cuenta el perfil de U2, que además de privilegiar los servicios sin costo, prioriza servicios con alta disponibilidad. En este caso el plan resultante fue idéntico al plan obtenido a partir de las preferencias del U1 (Fig. 9). Es importante destacar que los 5 servicios utilizados en este plan no tienen costo monetario asociado. En la Tabla 5 se pueden apreciar los resultados obtenidos, resaltando la maximización del impacto en la composición de servicios.

TABLA V
RESULTADOS DEL CASO DE ESTUDIO #2.

	Impacto con Preferencias U1	Impacto con Preferencias U2
Composición sin preferencias	-10	-10
Composición con preferencias	**-5**	**-9**

V. DISCUSIÓN

Los resultados obtenidos muestran que la utilización de preferencias logra obtener soluciones más acorde con los requisitos del usuario. Debido a cómo están compuestos los datasets y la preferencia por utilizar servicios sin costo, la cantidad de servicios necesarios para obtener una solución puede incrementarse. Esto tiene un problema particular, a mayor cantidad de servicios, mayor es la probabilidad de que alguno de estos falle durante la ejecución. Con lo cual, el usuario debe tener en cuenta tanto la disponibilidad de los servicios como la preferencia definida para evitar posibles problemas durante la ejecución de la solución.

A su vez, el enfoque permite la introducción de preferencias por parte del usuario, permitiendo privilegiar la utilización de algunos servicios sobre otros. Las preferencias pueden ser definidas de dos formas distintas. La primera de estas se realiza es a través de los atributos de los servicios definidos en la ontología (costo y disponibilidad). En este caso el enfoque calculará de manera automática un valor de preferencia teniendo en cuenta los valores de los atributos. La segunda consiste en definir un valor de preferencia para cada servicio desde la interfaz del usuario, permitiendo a éste privilegiar los servicios requeridos a su manera. Por ejemplo, si hay dos servicios del clima, y el usuario considera que uno de ellos es más preciso que el otro, puede darle una mayor preferencia.

Las preferencias asociadas a cada servicio son tenidas en cuenta durante la ejecución del algoritmo de planeación. El mismo, ordena la lista de servicios disponibles considerando en primer lugar los servicios con mayor valor de preferencia. Luego, recorre la lista en forma ordenada, comenzando por los mayores valores de preferencia, comprobando por cada servicio, si la inclusión de éste en la solución parcial puede conducir a la solución establecida por el usuario.

VI. CONCLUSIONES

En este trabajo se presentó un enfoque capaz de interpretar, planificar, componer y llevar a cabo la invocación de servicios web de manera automática. El enfoque fue evaluado con 2 casos de estudio, que incluyen tanto las de preferencias, como la omisión de las mismas. Los resultados obtenidos cuando se utilizaron preferencias del usuario mejoraron sustancialmente los planes generados. Las soluciones obtenidas incluyen un número mayor de servicios privilegiados por el usuario.

En ambos casos se pudo comprobar que la utilización del algoritmo AG UCPOP permite obtener un resultado más acorde a los requisitos del mismo. Sin embargo, los resultados obtenidos en las pruebas demuestran que en algunos casos el número de servicios puede incrementarse dependiendo de las preferencias del usuario.

Si bien el enfoque ha demostrado ser capaz de generar resultados más acorde a las preferencias del usuario, el mismo posee limitaciones. En primer lugar, el lenguaje utilizado para definir el estado inicial y final debe poseer la misma sintaxis que el utilizado por el algoritmo de planeación. En segundo lugar, las descripciones OWL-S para la composición no siempre pueden estar disponibles, y más aún, deben estar definidas sobre la misma ontología para que la base de conocimiento esté unificada. Por último, sólo se ha evaluado el enfoque propuesto con 2 casos de estudio.

Como trabajo futuro, se propone desarrollar un componente que permita definir los estados utilizando el lenguaje natural para, posteriormente, realizar la traducción hacia el lenguaje del algoritmo utilizado en el enfoque. Adicionalmente, abordaremos el problema de que un servicio falle. Si dado un plan, y en medio de la ejecución falla un servicio, se podría re-planear tomando un nuevo estado inicial que incluya los efectos de los acciones que se llegaron a ejecutar hasta el momento. Al re-planear habría que remover del modelo la acción respectiva al servicio que falló, para que el algoritmo de planeación no considere ese servicio.

Por otro lado, un trabajo futuro a incorporar en nuestro enfoque es el hecho de aprovechar la estructura de las restricciones provistas por AG UCPOP para definir presupuestos acotados dentro de los perfiles de los usuarios. Es decir, que un usuario tenga un monto disponible para invertir en el uso de servicios. Esto claramente impactaría en el plan generado por AG UCPOP. Finalmente, el objetivo es generalizar los resultados experimentales e inclusive evaluar otros atributos de calidad, como escalabilidad.

REFERENCIAS

[1] F. M. Mena, R. H. Ucan, V. U. Cetina and F. M Ramirez, "Web service composition using the bidirectional Dijkstra algorithm". IEEE Latin America Transactions, 14(5), 2522-2528. 2016.

[2] F. McCabe, D. Booth, C. Ferris, D. Orchard, M. Champion, E. Newcomer, H. Haas, "Web services architecture." W3C note. 2004.

[3] G. Rodríguez, A. Soria, M. Campo, "Artificial intelligence in service-oriented software design." Engineering Applications of Artificial Intelligence, 53, 86-104. 2016.

[4] E. Sirin, B. Parsia, D. Wu, J. Hendler, D. Nau, "HTN planning for web service composition using SHOP2." Journal of Web Semantics, 1(4), pp. 377-396. 2004.

[5] S. D. Weld, "An introduction to least commitment planning." AI magazine, 15(4), 27-27. 1994.

[6] G. Rodríguez, L. Berdun, Á. Soria, A. Amandi, M. Campo. "Un enfoque inteligente para asistir en la planificación de proyectos ágiles". In XV Argentine Symposium on Artificial Intelligence (ASAI). 2014.

[7] A. Monteserin, L. Berdún, A. Amandi. "Analysing the PDDL language for argumentation-based negotiation planning". In International Conference on Computational Science and Its Applications pp. 698-713. Springer, Berlin, Heidelberg. 2012.

[8] L. Berdun, A. Amandi, M. Campo, "An agent specific planning algorithm." Expert Systems with Applications, 39(5), 4860-4873. 2012.

[9] S. Chattopadhyay, A. Banerjee. "Qos constrained large scale web service composition using abstraction refinement". IEEE Transactions on Services Computing. 2017.

[10] F. Chen, R. Dou, M. Li, H. Wu. "A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing". Computers & Industrial Engineering, pp. 423-431. 2016.

[11] M. Cremene, M. Suciu, D. Pallez, D. Dumitrescu. "Comparative analysis of multi-objective evolutionary algorithms for QoS-aware web service composition". Applied Soft Computing, pp. 124-139. 2016.

[12] A. S. da Silva, H. Ma, M. Zhang. "Genetic programming for QoS-aware web service composition and selection". Soft Computing, 20(10), 3851-3867. 2016.

[13] Z. Wang, B. Cheng, W. Zhang, and J. Chen, "Q-Graphplan: QoS-Aware Automatic Service Composition with the Extended Planeación Graph." IEEE Access, 8, 8314-8323. 2020.

[14] L. Purohit, S. S. Chouhan, and A. Jain, "Dynamic Web Service Composition Using AI Planning Technique: Case Study on Blackbox Planner." In Social Networking and Computational Intelligence pp. 183-195, Springer, Singapore. 2020.

[15] N. Lin, U. Kuter, E. Sirin, "Web service composition with user preferences." In European Semantic Web Conference pp. 629-643. Springer, Berlin, Heidelberg. 2008.

[16] Y. Zhao, S. Wang, Y. Zou, J. Ng, T. Ng. "Automatically learning user preferences for personalized service composition". In 2017 IEEE International Conference on Web Services (ICWS) (pp. 776-783). 2017.

[17] H. Wang, P. Ma, Q. Yu, D. Yang, J. Li, H. Fei. "Combining quantitative constraints with qualitative preferences for effective non-functional properties-aware service composition". Journal of Parallel and Distributed Computing, pp. 71-84. 2017.

[18] D. Weld, J. Penberthy. "UCPOP: A sound, complete, partial order planner for ADL". In International Conference on Principles of Knowledge Representation and Reasoning pp. 103-114, 1992.

[19] V. Portchelvi, V. P. Venkatesan, G. Shanmugasundaram, "Achieving web services composition: a survey." Softw Eng, pp.195-202. 2012.

[20] D. Fensel, C. Bussler, C. "The web service modeling framework WSMF." Electronic Commerce Research and Applications, pp. 113-137. 2002.

[21] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, N. Srinivasan, "Bringing semantics to web services: The OWL-S approach." In International Workshop on Semantic Web Services and Web Process Composition pp. 26-42. Springer, 2004.

Sebastián Vallejos es actualmente miembro del ISISTAN Research Institute (CONICET-UNICEN), Argentina, estudiante de doctorado becado por CONICET (Consejo Nacional de Investigaciones y Técnicas). Se graduó de Ingeniero de Sistemas en 2016 (UNCPBA). Sus principales áreas de investigación son Procesamiento de Lenguaje Natural, Análisis de Redes Sociales y Programación Orientada a Objetos.

Leonardo Da Rocha Araujo es actualmente miembro del ISISTAN Research Institute (CONICET-UNICEN), Argentina, estudiante de doctorado becado por CONICET (Consejo Nacional de Investigaciones y Técnicas). Se graduó de Licenciado en Sistemas de Información en 2017 (UNIT, Aracaju, Brasil). Sus principales áreas de investigación son Minería de Texto y Arquitecturas de Microservicios.

Guillermo Rodríguez es actualmente miembro del ISISTAN Research Institute (CONICET-UNICEN), Investigador Adjunto de CONICET y profesor de UNCPBA. Se graduó de Ingeniero de Sistemas en 2010 (UNCPBA), y Doctor en ciencias de la Computación en 2014 (UNCPBA). Sus principales áreas de investigación son Desarrollo de Software Orientado a Servicios y Razonamiento Basado en Casos.

Luis Berdun es actualmente miembro del Instituto de Ingeniería de Software Tandil, Argentina, Investigador Adjunto de CONICET y profesor de UNCPBA. Se graduó de Master en Ingeniería de Sistemas en 2005 (UNCPBA), y Doctor en ciencias de la Computación en 2009 (UNCPBA). Sus principales áreas de investigación son Agentes Inteligentes, Algoritmos de Planeación, Gerenciamiento del Conocimiento.

Renzo Toscani obtuvo el título de Ingeniero de Sistemas en el año 2018 por la Facultad de Ciencias Exactas de Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA, Argentina).