Semantic web services composition in the astrophysics domain: issues and solutions

Thierry Louge, Mohamed Hedi Karray, Bernard Archimède, Zakaria Maamar, Michael Mrissa

HAL Id: hal-02356312
https://hal.science/hal-02356312

Submitted on 8 Nov 2019

Open Archive Toulouse Archive Ouverte (OATAO)

This is an author's version published in: http://oatao.univ-toulouse.fr/22777

Official URL: https://doi.org/10.1016/j.future.2018.07.063

To cite this version:

Semantic Web Services Composition in the astrophysics domain: Issues and solutions

Thierry Louge [a,*], Mohamed Hedi Karray [b], Bernard Archimède [b], Zakaria Maamar [c], Michael Mrissa [d]

[a] Université de Toulouse - UPS, Institut de Recherche en Astrophysique et Planétologie - IRAP, 57 Avenue d'Azereix, 65000 Tarbes, France
[b] Université de Toulouse, Ecole Nationale d'Ingenieurs de Tarbes ENIT, 47 avenue d'Azereix, BP1629, 65016 Tarbes Cedex, France
[c] Zayed University, College of Technological Innovation, Dubai, United Arab Emirates
[d] Université de Pau et des Pays de l'Adour, Avenue de l'Universite 64000 PAU Cedex, France

HIGHLIGHTS

- Automatic Semantic Web services composition.
- Composition of Web and non-Web services.
- Answer to specific requirements for Astrophysical services composition.
- User friendly approach.
- QoS assessment based on users' feedback.

ABSTRACT

Semantic Web services (SWS) have been hailed for their role in realizing the potential of composing services in the context of service-oriented computing. However, many application domains, such as astrophysics, require a major rethinking of how service composition should take place despite his potential. In this paper, we describe an approach for automatic, user-friendly SWS composition for the astrophysics domain that features certain particularities like correlated or optional inputs, aggregates, and inclusion of non-Web services. The approach selects the best SWSs using non-functional requirements along with users' previous experience feedback and is demonstrated using some real astrophysics services.

1. Introduction

Web Services Composition (WSC) and Automatic Web Services Composition (AWSC) are active topics in the Information & Communication Technology (ICT) community. In a short period of time, those topics made considerable progress with respect to many theoretical studies, but suffer from a limited number of concrete applications that could showcase the usefulness of Web services [1]. Many approaches tackle WSC from different perspectives for instance, communication protocols, discovery techniques, Quality of Service (QoS), security, and fault-tolerance [2]. In conjunction with all these perspectives, composition remains a cornerstone to the success of WSs as it spreads over several stages including discovery, selection, and execution.

To enable a fully-automated use of WSs, so, that the goals of the semantic Web are achieved [3], a semantic representation

of knowledge contained in these WSs is deemed necessary. This representation refers to ontologies. Ontology Web Language for Services (OWL-S) [4] and Web Services Modeling Ontology (WSMO) [5] are ontologies that offer concise representation of WSs and enrich their semantic description. Services described in OWL-S or WSMO are referred to as Semantic Web Services (SWS).

In a 2016 survey [6], a framework for service composition analysis was developed. The authors evaluated the relevance, significance, impact, and originality of several composition approaches reported in journals and conferences, along with their own knowledge of the domain. Twelve (12) approaches were selected and their analysis has led to two (2) major findings:

- Approaches based on Simple Object Access Protocol (SOAP) and REpresentational State Transfer (REST) are the most widely used (eight (8) out of twelve (12)). Only one approach was based on OWL-S, and is known as Simple Hierarchical Ordered Planner (SHOP)2 [7]. This shows that SWS compositions targeting real-cases along with valid results are scarce.

* Corresponding author.
E-mail address: thierry.louge@irap.omp.eu (T. Louge).

• Only three (3) composition approaches are dedicated to scientific applications.

In addition to these two findings, none of the approaches has been classified as end-user app remixers, which according to the framework's terminology would designate a novice user with non-technical background about how to compose services. Every approach targets technical users with a certain level of familiarity with service technologies

Taverna is an example of the approaches examined in the 2016 survey. It allows an IT-novice user to download and run a workflow of services, but modifying the workflow is not feasible. This lack of user support was already pointed in Bartalos and Bielikova survey [1].

Recently, astrophysics has been focusing on developing a dedicated Distributed Computing Infrastructure (DCI) called the Virtual Observatory (VO), which is the largest source of astrophysical data that would be made available to the astrophysicist community. Data interoperability in the VO is partially achieved using vocabularies like Unified Content Descriptors (UCDs)[1] and UTYPES[2] that allow sharing common formats. VO services may be discovered by querying VO service registries using keywords. The discovery leads to XML descriptions of services that finally can be run by a VO-compliant software. Many WSs (more than ten thousand (10.000)) are available through this infrastructure. Unfortunately, adapting existing service composition approaches to the VO is not straightforward. Indeed, astrophysics itself presents some specificities that make re-thinking service composition a must. The main difficulties are:

• VO architecture does not embrace existing technologies like Web Services Description Language (WSDL) and SOAP, but promotes its own formats and protocols; which make existing SWS composition approaches inappropriate.
• Most astrophysical services are atomic and stateless. A service may be called with a set of mandatory inputs that may be reinforced with non-mandatory inputs. During composition, a service is selected following the usefulness of its outputs and completeness of its input values. Therefore, the fact that whether a specific input is mandatory or not for a given service shall be known during the composition. Service inputs may sometimes be strongly connected in a way that a set of inputs from a service must come from a unique source.
• When a service is described as an output provider (e.g., temperature of a star) for an input (e.g., name of a star), it is not sure that the service will provide the output for every star. The service will only provide the output for a set of stars, those that have been observed by the instrument connected to the service. During composition, this implies that one can never be sure that a service's outputs will be provided. This depends on the overall context of the composition that consequently must be depicted to ensure the best guarantee of success.
• Another particularity of astrophysics is the specialized vocabulary (or jargon). General terms can be broken into more accurate sub-terms, themselves sometimes described in ways that may differ from one service to another (e.g. "K-band mag" designates the same quantity than "Magnitude in K" or "Magnitude in the band K"). Same applies to the units and data formats. While the VO provides an interoperability layer for addressing heterogeneous data, the current content of services' descriptions and content greatly depends on the context of the composition. This is exemplified with the

fact that it is sometimes necessary to adapt the VO formats themselves according to the subfields to be described such as astrochemistry in VAMDC [8].

Ensuring an automatic SWS composition for astrophysics requires a new way for addressing the above difficulties through the use of a specialized ontology. In a previous work, we developed an ontology for astrophysical services known as Astrophysical Services Ontology (ASON).[3] This ontology gives the opportunity to access astrophysical WSs and analytical services through an OWL-S based ontological description, which reduces the complexity of service composition to a SWS composition one. The remainder of this paper is organized as follows. Section 2 provides an overview of recent research work on service composition and briefly presents ASON. Section 3 exposes the concrete application of services composition in astrophysics. Algorithms, ontology integration, and QoS concerns are also presented in this section. Concluding remarks and future work are presented in Section 4.

2. Related work

This section consists of four (4) parts. After review of SWS composition, the specificities of astrophysical services are presented. The last two (2) parts of the section present the ontology used for astrophysical SWS description, and the motivations underlying our work.

2.1. Semantic web services composition

The semantic Web is an ongoing evolution of the existing Web architecture of Berneers-Lee [3]. This evolution exposes the knowledge of WSs in terms of data and rules helping understand these data.

SWS roots [9] refer to languages that express semantics such as DARPA Agent Markup Language (DAML-S) (upon which OWL-S is developed) and use logic programming languages to perform reasoning. While WS composition has seen some success, SWS composition remains heavily theoretical; SWS have not yet been widely adopted [10] despite OWL-S and WSMO. A reason is the difficulty of properly annotating existing WSs so they are turned into SWS [10,11]. This is corroborated by Lemos et al.'s survey [6] that only refers to one SWS-based approach.

Ongoing research either addresses specific aspects of the composition process or the entire process. This process includes different stages such as selecting services based on Quality of Service (QoS). In Zhao et al. [12], non-functional parameters are hierarchized using fuzzy logic and possible compositions are sorted based on this hierarchy.

iServe [10] approach addresses WSs discovery by sharing the semantic annotations associated with every service. Those annotations are reachable using different means such as a dedicated RESTful API, Resource Description Framework (RDF) triples query, or SPARQL Protocol and RDF Query Language (SPARQL) reasoning. Thus, iServe acts as a service registry through semantic annotations. Another use of SPARQL for WS discovery is reported in [13], focusing on the expression of post- and pre-conditions of OWL-S services. This work assumes that a (may be remote) registry is available for accessing services operations and building graph patterns expressing pre-conditions and effects of services. Each OWL-S ontology must also share the same reference of terms to express services descriptions and goals. The work of Sbodio et al. does not address the "minimal sufficient conditions" (what is absolutely needed for a service to run), that may differ from services inputs and preconditions. QoS aspects are not addressed, either.

Puttonen et al. [14], also, use SPARQL and OWL-S to compose and invoke SWS. Their approach achieves a goal state that fulfills the composition's requirements. They adopt forward-chaining for service selection, going from available pre-conditions at the start of the composition (the initial state) and exploring every single available path from this initial state to the goal state. Unfortunately, Puttonen et al. do not reuse previous compositions, nor the impact of QoS on compositions.

Reusing composition history is examined in [15], where a backward-chaining composition algorithm (referred to as "from right to left") for single output, multiple input services, is discussed. The main limitation is the difficulty to provide efficient domain ontology for WS description. QoS is not addressed as well. There is a good number of QoS-based SWS composition approaches. Rodriguez-Mier et al. [16] propose a forward-chaining and backward pruning composition algorithm, based on iServe for service discovery. Every possible combination of services to fulfill the goal is generated in a first step. Then, a second algorithm is run over those combinations to find the minimum-cost combination. The criteria for establishing the cost may vary (e.g., number of services and QoS parameters).

Bansal et al. [17] discuss SWS composition by developing a Prolog-based discovery and composition, the repository of facts deriving from the semantic descriptions of services expressed in Universal Service-Semantics Description Language (USDL) [18]. The discovery of services is based on mapping between those services USDL descriptions, description of the composition requirements, and WordNet [19] (or a domain-specific ontology). WordNet or domain ontology serves as the reference for defining concept meaning. The choice of services relies on measuring the "degree of centrality" representing the number of service providers that a service provider is in contact with. This implies a semantic description providing the information of links between services; the higher the centrality degree is, the better the service rating becomes. The resulting compositions are stored like OWL-S documents. This approach does not address the execution phase of composition; its result is the composition itself and not the result of the execution.

Recent works on automatic composition of services highlight that approaches for automated RESTful services composition remain limited. Besides, providing such a composition for heterogeneous services requires a universal service description, and that is a challenge [20,21].

Astrophysics is a good real-world testing domain for such concerns, as automated service composition requires mixing Web and non-Web services. VO services are RESTful-like, meaning that they use the Web for communication but without fully endorsing all the REST architecture specification. The composition requires to be fully automated, i.e., no-human assistance and to consider users' feedback and composition history during QoS evaluation. This QoS focuses on user-centric parameters, such as service reputation, specialization, and completeness based on users' feedback rather than on response time or service availability. As a consequence, this QoS highlights factors that may lead to different QoS perceptions by different users, which is not usually the case [20].

Our work is one step towards service composition in the astrophysics domain. It encompasses all the composition steps (discovery, composition including QoS parameters, and orchestration). No specific language or format is imposed to express requirements and discover services and composition refers to a single ontology for service registry and description.

2.2. Why astrophysical services are special?

Before delving into the details of service composition, we discuss why astrophysical services are special. Those specificities mainly reflect that astrophysical quantities (e.g., measures that constitute most of the outputs of astrophysical services) are not isolated concepts in an ontology. They rather are sets of concepts tied together by relations, expressing units and formats under which a service accepts or expresses the quantities it provides or needs to be executed. Also, combination of services' inputs must reflect unusual conditions such as correlated information (i.e., services' inputs must come as outputs from the same previous service in the composition).

SWS definition and input discovery problem can be found in Rodriguez-Mier et al. [16]:

"A Semantic Web Service (SWS, hereafter "service") can be defined as a tuple $w = \{In_w, Out_w\} \in W$ where In_w is a set of inputs required to invoke w, Out_w is the set of outputs returned by w after its execution, and W is the set of all services available in the service registry.

Each input and output is related to a semantic concept from an ontology O (In_w, $Out_w \subseteq O$). [...]. Given a set of concepts $C \subseteq O$, the input discovery problem consists of finding a set of relevant services $W = \{w_1, \ldots, w_n\}$ where $w_i = \{In_{w_i}, Out_{w_i}\}$ such that $\forall w_i \in W$, $C \otimes In_{w_i} \subseteq In_{w_i}$".

In those definitions, O is an ontology and "$C1 \otimes C2$" is an operation that returns the concepts in C2 match C1, with C1 and C2 being two sets of concepts.

Similar definitions can also be found in Definitions 6 and 11 from the work of Bartalos et al. [1]. In the case of astrophysics, things are quite different. Expressing input conditions and provided outputs for a service means expressing the astrophysical quantities together with their relevant format and unit. Therefore, $C \otimes In_{w_i} \subseteq In_{w_i}$ does not fully describe the conditions to meet during input discovery. Mechanisms exist in ASON, briefly described hereafter, that allow to express those units and format conditions. The first mechanism is aggregation. Formal definition of aggregation for ontologies can be found in the work of Severi et al. [22]. Aggregates in ASON ties together astrophysical quantities, formats, and units using the following relations:

- IsCombinedToUnit, that links aggregate with its unit noted $ICU(G_i, U_j)$, $G_i \in ASON$ being the aggregate, $U_j \in ASON$ the unit;
- IsCombinedToParam, noted $ICP(G_i, Q_k)$ that links aggregate with its astrophysical quantity, $G_i \in ASON$ being the aggregate, $Q_k \in ASON$ the astrophysical quantity;
- And, IsCombinedToFormat, that links aggregate with its format noted $ICF(G_i, F_l)$, $G_i \in ASON$ being the aggregate, $F_l \in ASON$ the format.

The description of an astrophysical service is then a triple $w = < GIn_w, GOut_w, A$ with $GIn_w >$ and $GOut_w$ being the aggregates describing services' inputs and outputs, respectively and

$A = ICU(G_i, U_j), ICP(G_i, Q_k), ICF(G_i, F_l) \forall G_i, U_j, Q_k, F_l \in ASON$

being the set of relations identifying units, quantities and formats of the aggregates. Rodriguez-Mier et al. [16] discuss when a service in a composition becomes invokable:

"If $C \subseteq O$ is the set of available input concepts, then a service $w = In_w, Out_w$ is invokable with C if $C \otimes In_w = In_w$, i.e., there exists a full matching between the available inputs and service inputs".

Here are two more criteria encountered when dealing with astrophysical services:

- Inputs have different status that can be either mandatory or optional. Only mandatory inputs for a service are to be matched, whereas optional inputs improve the quality of the outputs.
- Some inputs are linked to each other in a way that they must be provided by outputs of the same service. We call those inputs "correlated inputs". They indicate measures that only make sense when coming from the same source, because

they need the exact same observing conditions, for example. For instance, when a service needs a value and the error bar related to this value. Both value and error bars become meaningless if they do not come from the same source.

At the top of Fig. 1, extracted from [16], service w6 of layer L2 may be used as long as "ISBN", "Address" and "AuthCode" are provided. At the bottom of the same figure, we make the assumption that some of w6's inputs are correlated. "I1" square indicates that if "ISBN" and "AuthCode" were correlated inputs, then w6 could not be used in the same composition. In the meantime, for w9 if we suppose that "BookingNum" and "Business" are correlated ("I2" square), w9 may still be used.

ASON defines two relations to address these concerns; "HasCorrelatedInput" (HCI) linking the service to an aggregate that indicates correlated inputs, and "IsCorrelatedWith" (ICW) that indicates which inputs are correlated with each other. Each correlation of inputs is registered through an aggregate. Those aggregates are linked to the service through HCI and to the inputs through ICW. Those inputs are themselves aggregates linking quantities, units and formats.

In order to take into account these specific characteristics of astrophysical services, we propose the tuple defined below:

Definition 1.

$$w = \{MGIn_w, OGIn_w, GOut_w, A, R\} \tag{1}$$

with:

- $MGIn_w$ being the aggregates defining mandatory inputs for the service ;
- $OGIn_w$ being the aggregates defining optional inputs;
- $GOut$ being the aggregates describing the services outputs;
- $A = ICU(G_i, U_j), ICP(G_i, Q_k), ICF(G_i, F_l) \forall G_i, U_j, Q_k, F_l \in ASON$ being the set of relations identifying quantities, formats, and units of the aggregates;
- And, $R = ICW(G_i, \ldots, G_j) \forall G_i, G_j \in ASON$ being the set of input aggregates correlated to each other.

Conditions for invoking a service are then:

If $G \subseteq ASON$ is the set of available input aggregates, then a service $w = \{MGIn_w, OGIn_w, GOut_w, A, R\}$ is invokable with G if:

- $G \otimes MGIn_w = MGIn_w$;
- $\forall ICW(G_i, \ldots, G_j) \in R; G_i \ldots G_j \in GOut_{w_x}, w_w \in ASON$.

2.3. Overview of astrophysics services ontology

The majority of existing WSs for astrophysics are registered in the International Virtual Observatory Alliance (IVOA) architecture. This architecture is shown in Fig. 2 divided into blocks [23].

The upper block comprising users and computers (which represent manual or automatic use of the OV) interacts with "USER LAYER". This user comprises the means for accessing the OV, whether by WS, dedicated software, or scripts. The lower block presents providers who interact with the "RESOURCE LAYER" to make their data available. The service access protocols (how to use the services) appear in the block on the right. Simple Spectrum Access (SSA), Simple Image Access (SIA), and Table Access Protocol (TAP) are examples of protocols available in IVOA specifications. The central block concerns the elements supporting this architecture, such as data models and semantics (which refers to IVOA semantics definition, not related to the Semantic Web). IVOA data models are schemas for describing metadata associated with observational or theoretical data. These patterns may change over time depending on the feedback from service providers and their specific use cases. The distinction between data model and access

protocol is sometimes blurred. Thus, SSA is both a protocol and a data model.

Our SWS composition for astrophysics relies on ASON ontology, detailed in a previous work [24]. It offers universal services description specification, in the sense that heterogeneous services such as Web and non-Web based services can be described, despite their technical differences (e.g., RESTful *vs.* WSDL).

Aladin[4] is an example of a non-Web based service that may be part of services composition together with Web-based services. Aladin is a desktop-based application that can be automated through the use of custom scripts, in order to provide automatic analysis and display of astrophysical data. Those scripts are not called through the Web, and must be installed in the machine hosting the service composition system. They may then be used as composition components, just as any other service by the approach presented in this paper. ASON structure may be an OWL-S extension and the details are out of the scope of the current paper; contrarily, a brief overview of the ontology is provided below.

ASON consists of two (2) modules: Generic Ontology for Services (GEOS) that is built upon OWL-S and a thematic module. The thematic module describes the knowledge elements coming from the application domain, that is why we called this module ASTRO-THEM. Those elements are mainly useful for expressing a shared conceptualization of services' inputs and outputs. By using the two modules, we link the service capacities expressed in GEOS to the domain elements embedded in the thematic module.

GEOS describes the functional aspects of services by taking care of the grounding of services using the notion of "protocol". The VO architecture offers several protocols for accessing VO services. The most widely used are SSAP (Simple Spectrum Access Protocol), SIA (Simple Image Access) and ConeSearch. Those protocols are VO-specific, and the orchestration and execution phases of the composition must ensure that any combination of services using those protocols may be used. A protocol expresses a common set of mandatory input information and a common query formulation for a set of services. Another important concept for GEOS groundings is the "QuerySoftware". It turns the input information for a service into concrete message sent to the service using the theoretical service description (protocol, inputs, URL, outputs), and turns the service response into concrete information usable by other services in the composition.

GEOS is the result of several years of work dealing with astrophysical services semantic expression. Nevertheless, the mechanisms in GEOS may be useful for other scientific fields running into the same challenges than astrophysics in the process of going from WSs to SWS: semantics divided into separated aspects, inputs that may be mandatory or optional and with sub-sets that may be correlated, and a grounding integrated inside a pre-existing DCI such as the VO.

ASON contains services, so that it acts both as an ontology for service description and a service registry. The classes in ASON describe generic quantities, such as "Magnitude", which is a measurement of brightness for stars that can be specialized in many ways. The individuals belonging to the classes are the elements inside the ontology that express the most detailed description of a measurement. Fig. 3 represents the main components of ASON architecture. In its current state, ASON describes eleven thousand one hundred thirty five (11135) services, for fourteen thousand one hundred ninety-nine (14199) different aggregates.

The description of services in ASON comes from an automatic method for ontology construction. This method, which is currently being published, allows the development of an ontology from short, unstructured text such as the astrophysical quantity
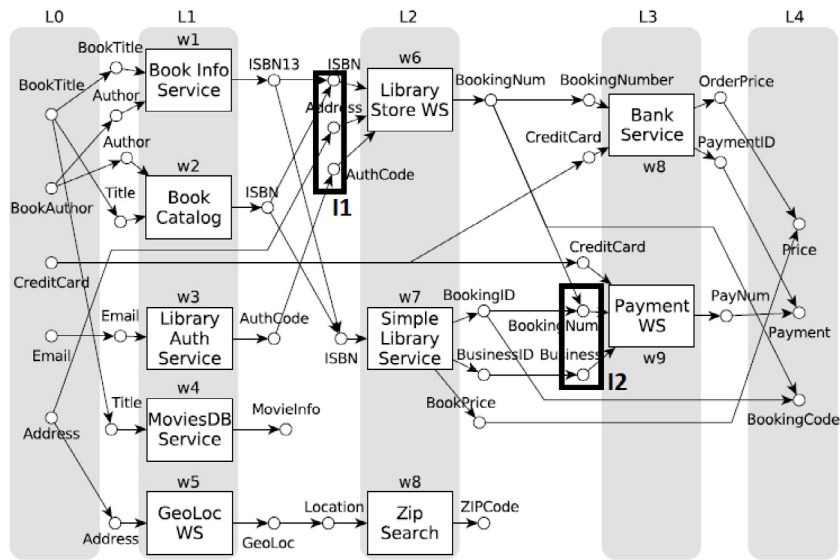
---

4 http://aladin.u-strasbg.fr/.
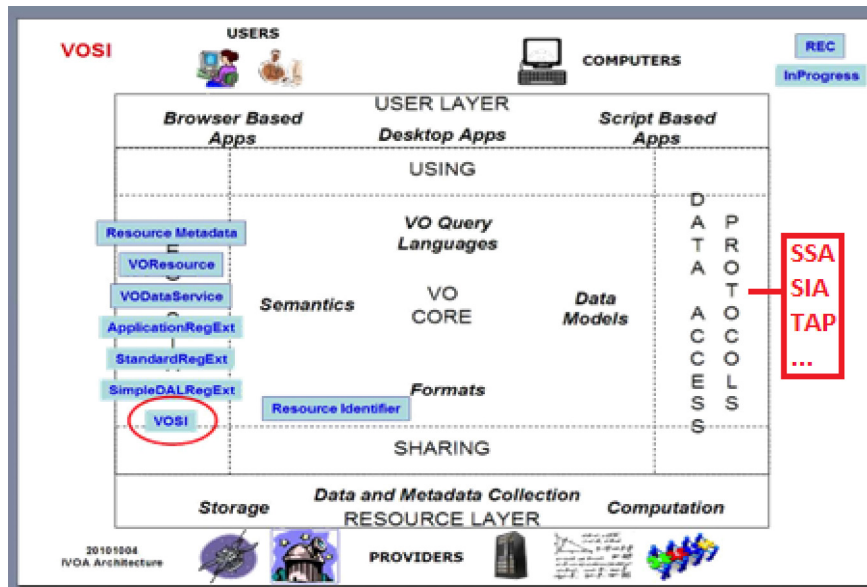
Fig. 1. Correlated inputs example.

Fig. 2. IVOA architecture for astrophysical Web services.

descriptions contained in the description of astrophysical services. This ontology constitutes the core of the thematic module. The GEOS module is populated by the analysis of XML documents describing services. Information about the protocol used by the services, query and test addresses are extracted from XML documents and translated into OWL according to the relationships and concepts proposed in GEOS.

2.4. Motivations

Lemos et al.'s survey identified SHOP2 as the only approach dealing with SWS and Taverna as the only approach dealing with scientific workflows. To the best of our knowledge, there is a lack of approaches for SWS composition in astrophysics apart from certain specific software specifically designed for specific astrophysical instruments like scientific gateway for the instrument CTA [25]. A more generic SWS composition in this field would be highly appreciated by the community.

Delivering a user-friendly interface is mandatory for SWS composition in astrophysics to achieve its full potential. This means:

- Providing the user with an easy way of expressing requirements for the composition;
- Ensuring that the composition returns results;
- And, executing the resulting compositions, displaying the results, and allowing quick re-composition of services by adjusting the necessary services' selection criteria.

Expressing the requirements of a composition in an easy way for users implies reifying the natural language with the semantic description of services' outputs and inputs. It is necessary to have a more flexible system than a simple keyword matching that would not recognize unregistered keywords in a certain repository. A keyword-based system could lead to the use of particular habits as a "de facto" standard (e.g., using "k-band magnitude" instead of "magnitude in band k" or *vice-versa*).
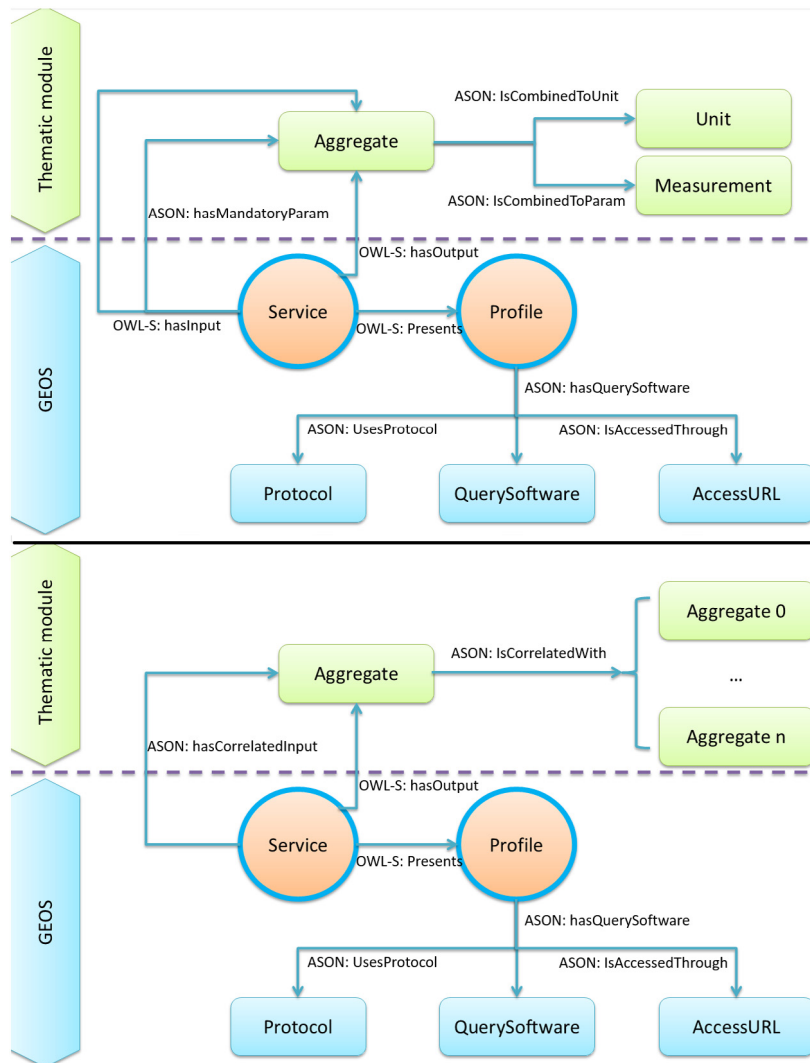
Fig. 3. ASON structure overview.

Ensuring the quality of a composition is critical for astrophysics. Since every service has its own specificities, there are two important parameters to consider: service specialization regarding some specific information and current availability of given information, inside a given service for a given input. Service specialization may only be confirmed by the user, so that there is a need for providing the user with the possibility of rating the service.

Ensuring that the services elected for a composition are the ones that will provide useful information means that taking the execution phase into account in our approach is crucial. Indeed, when a service feeds the workflow with relevant data, this is logged in a history file, so that the same service for the same context (inputs values, and requested output information) may be privileged over services for which the relevancy in the context is unknown. A list of every service returning usable information when used in conjunction with one another is, also, produced without input consideration. This list is composed of parts of workflows (or entire workflows, when every service provides useful output). Those lists are tested during compositions, so that previously-used services in conjunction with one another and providing useful outputs will be preferred. This materializes the fact that such services partially share the same context.

Executing the selected composition is the final step of the composition, and the most important from the end-user perspective.

It is the phase where results are displayed and lessons learned about services content and quality. In the composition presented in this paper, the production and display of the results of the workflow is also the time where the user may change the criteria for services selection; try different combinations or even running every possible composition.

Scalability and response time are not a challenge, despite the huge number of services described in ASON. The composition is designed to help astrophysicists in their research, and several dozens of minutes or sometimes hours of processing (for testing every combination of services possible, in large workflows) are not a concern in the context. Section 3 exposes how our approach handles the difficulties listed in the introduction and achieves the goals outlined above.

3. Astrophysical services composition

Fig. 4 shows the components of our service composition. Green elements are accessible by the end-user, through dedicated Web interface. The composition for astrophysical services uses ASON both as a service registry and as a semantic description repository for services. This is a difference with other semantic composition approaches such as "native" use of OWL-S [4], iServe [10] or more recent approaches [15], that rely on ontology alignment and service registries.
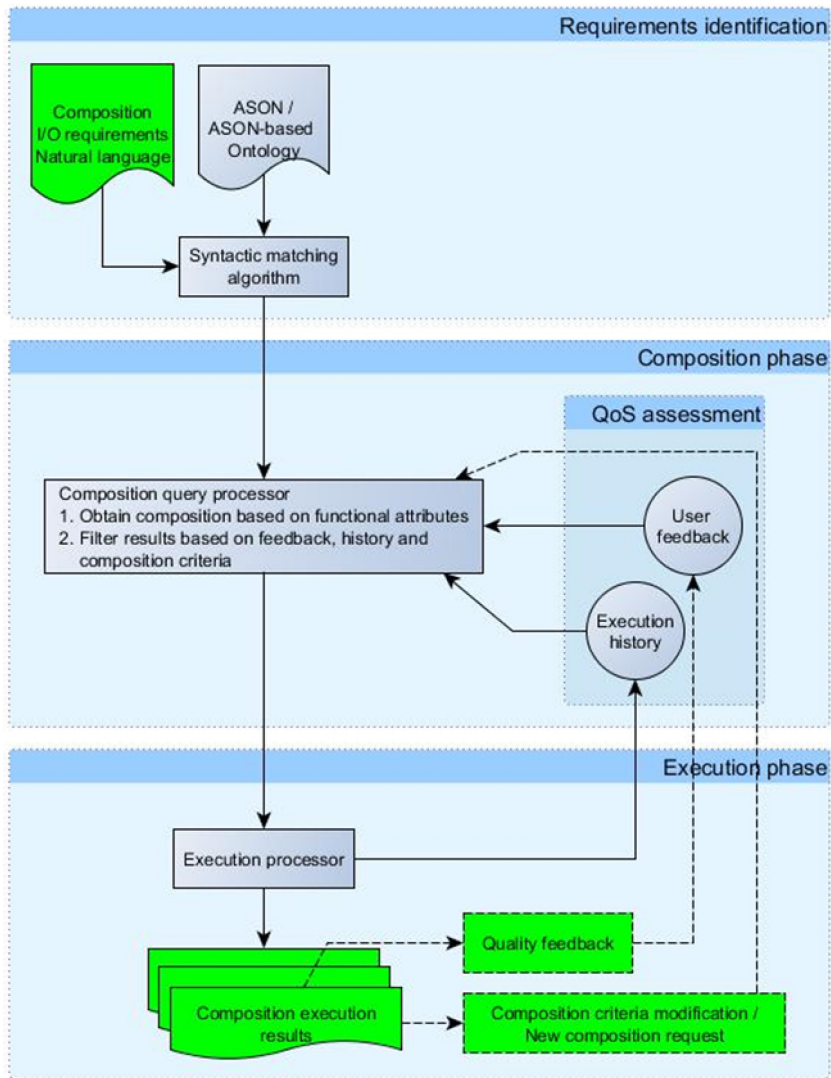
Fig. 4. Components of the composition.

The ontology alignment process is complex, and subject to errors or uncertainties. Moreover, it imposes a stage of discovery of ontologies to be compared, and also of external sources likely to distinguish ambiguous cases. Same considerations apply when bringing semantics to existing service registries. It is, then, necessary to disambiguate the concepts between the semantics applied to the different registries, in order to interoperate services.

ASON's dual nature of global ontology and directory of services reduces the alignment problem to the browsing of ASON's content. As a consequence, the composition algorithms for astrophysics essentially consist of ontology browsing for service discovery, composition and orchestration.

3.1. Requirements identification phase

User requirements concerning inputs and outputs are expressed without constraints concerning the terms that may be used. No specific keywords are required, and there is no mandatory order of words (e.g. "Magnitude in JK band" is as good as "J-K band mag"). Fig. 5 shows an example of user requirements (heliocentric radial velocity, field chart…) together with a subsample of results on the green rounds. We can see that "heliocentric radial velocity" is matched with its exact counterpart in the ontology when it exists, and also with the closest non-exact matchings available if

they are close enough. In that case, "heliocentric radial velocity (cz) optical measurement" is the closest non-exact matching, and close enough to be elected as a relevant answer.

Services' inputs and outputs are represented by individuals in ASON (instances of concepts in the ontology). The first step of service selection is to identify individuals that best match the natural language expression of the composition requirements. The identification of relevant individuals is based on a syntactic matching between I/O requirements given by the user and annotations of individuals. A similarity value is measured, that expresses the similarity between each description of I/O in the composition (description of information in natural language) and each annotation of individuals (description of information in natural language). The similarity value measurement algorithm, presented hereafter compares both descriptions. The following are some elements of notation for the similarity value measurement algorithm:

- Dx and Dy are set of words W, $|Dx|$ is the number of words in Dx;
- $Wn(Dx)$ is the n word contained in Dx, $Wp(Dy)$ is the p word contained in Dy;
- $|Wn(Dx)|$ is the number of characters composing $Wn(Dx)$, $|Wp(Dy)|$ is the number of characters composing $Wp(Dy)$.
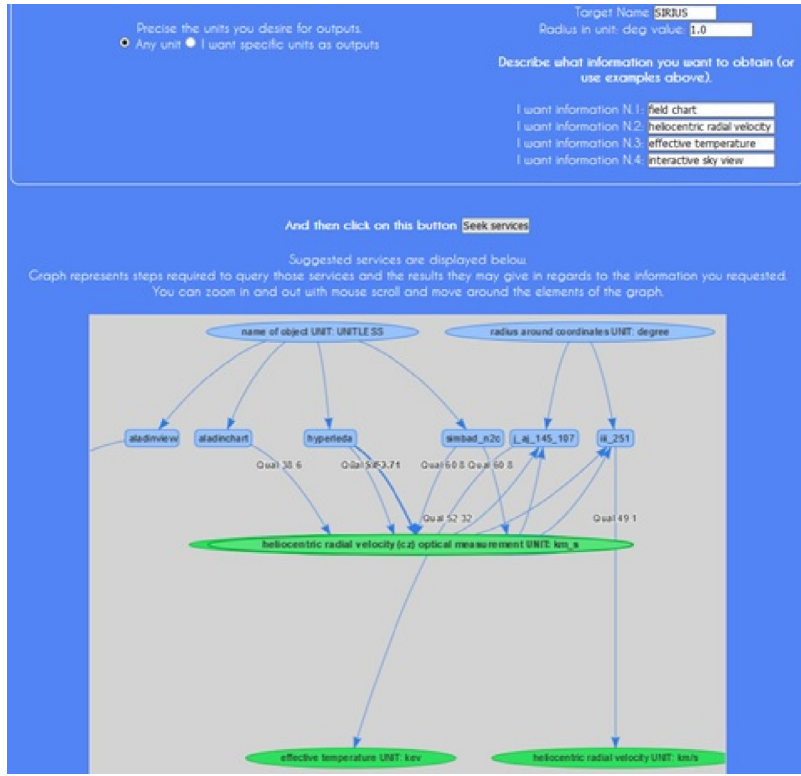- And, $S(Dx, Dy)$ is the similarity value between Dx and Dy.

Fig. 5. Expression of requirements, corresponding results.

The values for the similarity matrix are based on similarity measurements between each pair of words of each pair of descriptions, produced as follows:

$Lev(Wn(Dx), Wp(Dy))$ is the Levensthein distance [26] between $Wn(Dx)$ and $Wp(Dy)$.

The Levensthein distance between two strings is the minimum number of single-character modifications needed to make one of the two strings equal to the other. The modifications include replacement, deletion, or addition of characters.

$L(Wn(Dx), Wp(Dy))$ is the normalized Levensthein distance between $Wn(Dx)$ and $Wp(Dy)$.

$$L(Wn(Dx), Wp(Dy)) = 1 - \left(\frac{Lev(Wn(Dx), Wp(Dy))}{max(|Wn(Dx)|, |Wp(Dy)|)}\right) \quad (2)$$

$J(Wn(Dx), Wp(Dy))$ and $JW(Wn(Dx), Wp(Dy))$ are the Jaro and Jaro–Winkler distance [27] between $Wn(Dx)$ and $Wp(Dy)$, respectively.

The Jaro distance between two strings is calculated based on the number of matching characters between two strings, and the distance in which they are found relatively to the number of characters in the string. The "Jaro–Winkler" version of this measure gives better scores for strings in which a beginning set of characters are matching (emphasis on prefixes).

Those three distance measurements Levensthein [26], Jaro [27], and Jaro–Winkler [27], cover most of the cases that are found when dealing with domain-specific vocabulary like astrophysics. Indeed, it is necessary to separate cases when two strings describe close quantities using different places for words ("johnson b-v colour index" compared to "b-v (johnson) color") that are best matched with Jaro distance. Some descriptions differ in ways that are best matched with the Winkler version of this measure ("J-K mag" and "J-K band magnitude"). Finally, the Levensthein distance is suitable for matching descriptions that are close to each other, but where the words are in a different order and not expressed in the exact same way ("magnitude in J-K band" and "Band JK magnitude"). The

similarity value $S(Wn(Dx), Wp(Dy))$ is then:

$$S(Wn(Dx), Wp(Dy)) = max($$
$$L(Wn(Dx), Wp(Dy)),$$
$$J(Wn(Dx), Wp(Dy)),$$
$$JW(Wn(Dx), Wp(Dy)) \quad (3)$$

The trust value $T(Wn(Dx))$ is the maximum matching score for a single word Wn in Dx with every word from Dy, considering the normalized Levensthein distance, Jaro and Jaro–Winkler:

$$T(Wn(Dx)) = max(S(Wn(Dx), Wp(Dy))); p = 0..|Dy| \quad (4)$$

Finally, the similarity value $S(Dx, Dy)$ is the sum of every trust value from words in Dx divided by the number of words $|Dx|$. To ensure that the similarity measure is symmetric, $|Dx| = |Dy|$ is necessary. The shorter description is, therefore, filled with dumb word instances, "###" that will not match any other word with sufficient score to introduce any noise.

$$S(Dx, Dy) = \frac{\sum_{n=0}^{n=|Dx|}(T(Wn(Dx)))}{|Dx|} \quad (5)$$

For illustration purposes, let us consider the following descriptions:D1 = "johnson b-v colour index" and D2 = "b-v (johnson) color". After every measure for each pair of words: $S(D1, D2) = 0.72$.

For each I/O in the composition requirements, one individual in the ontology is identified. This individual is the one with the higher similarity value between one of its annotations and the considered I/O (i.e., S(individual annotation, considered I/O). Those individuals in the ontology will be passed to the ground work algorithm depicted in Table 1, either as input or outputs for the composition.

For some I/O, only weak similarity values can be found. A trigger is to be defined, which we set to 0.3 under which the requirement is considered like having no counterpart in the ontology. In the case

of an output, that means that this output will not be generated by the composition process. In the case of an input that means that this input will not be used during the composition process.

3.2. Service selection and workflow composition phase

In our approach, the selection of services' ensure that every input information needed by a service is provided. This is done by exploring the predecessors (a predecessor for a service is every service that outputs match with its inputs), and their own predecessors until a decision is made to include the chain of relevant predecessors in the composition. This shows that selection and composition take place concurrently. Composition requirements (the composition output values that the user wants to obtain) and composition inputs are expressed by the corresponding individuals in the ontology. All the possible compositions are generated, based on the composition inputs and resulting in the composition outputs. First, pre-existing DCI (like the VO) may provide protocols for accessing data. Those protocols federate services that share the same set of inputs (a set of coordinates on the sky expressed in decimal degrees, a radius around these coordinates also expressed in decimal degrees for astrophysics). Using a forward-chaining algorithm will lead to selecting every DCI-compliant service at once, when corresponding input values are disposable during the composition process. That is why a backward-chaining algorithm is used. This algorithm will, only, determine the usability of services that are identified as providers for relevant outputs in the composition Services with incomplete chain (with insufficient inputs to be used) will be excluded from the composition, whereas services with sufficient inputs will form the final composition chains. Service composition is determined from a knowledge base that is a version of ASON translated into Powerloom[5] language.

Algorithms 1 and 2 show the groundwork and composition algorithms, respectively. On the one hand, the groundwork algorithm's inputs are the requested provided information (the individuals resulting from the requirements identification phase), together with relative units and formats. If no specific unit and/or no specific formats are given, then every unit/format available is considered properly matching. The output from this algorithm is a list of services that provide the required outputs for the composition (identified individuals with relevant units and formats, line 3 of the algorithm), plus an update of the knowledge base (lines 6 to 9). The groundwork and composition algorithms implement the requirements deriving from Definition 1. The groundwork algorithm queries the ontology to select potentially relevant individuals for a given set of requirements. As a reminder, requirements are expressed as per Fig. 5.

On the other hand, the composition algorithm is run over the knowledge base that the groundwork algorithm updates, and with the service list generated by this algorithm. The output of the composition algorithm is a file containing the list of services that satisfy users' requirements (updated in lines 16, 17 and 23 of the algorithm) with their associated parameters (updated in line 27). The list of services includes every composition available for those requirements. It is a directed, acyclic graph which unweighted edges.

In some cases, it may happen that services registered in the composition file after running the composition algorithm are not relevant for the actual composition. This happens in a very specific case, when a service has multiple predecessors (identified through use of SeekPredecessors algorithm) and some of those predecessors can be used in the composition, whereas others cannot and the usable predecessors do not fulfill the inputs' requirements for the service. SeekPredecessors Algorithm ensures that if correlated

[5] https://www.isi.edu/isd/LOOM/PowerLoom/documentation/documentation.htmls.

Input: $W = \{(I0, U0, F0)...(Ix, Ux, Fx)\}$ Individual, Unit, Format: requirements of results from the composition; $G = \{(I0, U0, F0, V0)...(Ix, Ux, Fx)\}$ Individual, Unit, Format: information provided by the user

Output: S list of services ; KBR update

```
1  S = {}
2  foreach (I, U, F) in W do
3  │  S ← S + getServicesProviding(I, U, F)
4  end
5  foreach (I, U, F) in G do
6  │  Available(I)
7  │  HasDisposableUnit(I, U)
8  │  HasDisposableFormat(I, F)
9  end
```

Algorithm 1: Groundwork algorithm

Input: S list of services
Output: File F containing an ordered list of services that fulfills composition goals, with inputs and outputs ; File Fp containing services parameters (URL if any, and QuerySoftware)

```
1   foreach s in S do
2   │  P = SeekPredecessors(s)
3   │  if P != {0} then
4   │  │  S ← S + p
5   │  end
6   │  MP = GetMandatoryParams(s)
7   │  OS = GetOutputs(s)
8   │  IS = GetIntputs(s)
9   │  foreach (I, U, F) in MP do
10  │  │  if (Available(I),
11  │  │  HasDisposableUnit(I, U),
12  │  │  HasDisposableFormat(I, F)
13  │  │  ∀(I, U, F) ∈ MP then
14  │  │  │  Write(F,(I,U,F))
15  │  │  │  foreach (I,U,F) in IS do
16  │  │  │  │  Write(F,(I,U,F))
17  │  │  │  │  Write(F,s)
18  │  │  │  end
19  │  │  │  foreach (I, U, F) in OS do
20  │  │  │  │  Available(I)
21  │  │  │  │  HasDisposableUnit(I, U)
22  │  │  │  │  HasDisposableFormat(I, F)
23  │  │  │  │  Write(F, (I, U, F))
24  │  │  │  end
25  │  │  │  QS = GetQuerySoftware(s)
26  │  │  │  URL = GetUrl(s)
27  │  │  │  Write(Fp, QS, URL)
28  │  │  end
29  │  end
30  end
```

Algorithm 2: Composition algorithm

inputs are found for a service, then they really come from the outputs of one single service.

When a service is put apart from the composition, it is necessary to verify whereas its predecessors provide any useful information for the rest of the composition, or not. If they do not provide any information for another service in the composition, then they become irrelevant. This is detected and resolved by EliminateUselessServices algorithm, used in the QoS assessment phase.

Input: Service *s*
Output: *P* = {*p*} list of predecessors for s
1 CI = GetCorrelatedInputs(s)
2 NCI = GetNonCorrelatedInputs(s)
3 **if** *CI != {}* **then**
4 | Pred = getServicesProviding(*CIn, n = 0..|CI|*)
5 | P ← Pred
6 **end**
7 **if** *NCI != {}* **then**
8 | Pred = getServicesProviding(*NCIn, n = 0..|NCI|*)
9 | P ← P + Pred
10 **end**

Algorithm 3: SeekPredecessors algorithm

3.3. Quality of service assessment

The QoS selection of services for the best possible composition is automated in our approach (algorithm 4).
Its input are:

- U = {(I0,V0)…(In, Vn)} being the couples (Input, Value) given by the user;
- File F from composition algorithm;
- File H1 containing the history of precedent compositions;
- File H2 containing ratings of the outputs for services on the form (service, output, quality);
- And, file H3 containing previous compositions input values for compositions during which the service has yielded usable results on the form (service, input, value)

W(H1), W(H2), W(H3) in algorithm 4 are the weights for the quality assessments given by H1, H2, H3 content respectively. Algorithm 4 is based on the user feedback and execution history. The execution history keeps track of two (2) parameters:

- The composition input values for compositions during which a service produced usable result;
- And, a list of every service in the workflow that produced a usable result.

Those two parameters (both saved inside a file in line 32 of algorithm 4) have different meanings. The first states that a service candidate for a composition is known for providing a usable result when used in a composition with a given set of inputs values for the composition. Those input values may only be the set of inputs provided by the user, as the QoS algorithm elects services before the execution phase and therefore only accesses user-given input values. The first parameter therefore expresses that, for a given set of inputs values for the composition, a service has been used inside the composition and provided useful results (this parameter is tested in lines 16–22).

The second one expresses that, for a previous composition, a set of services used together (and not necessarily chained) have produced usable results. This indication means that those services are likely to share a common context (e.g., observation of galaxies). In algorithm 2, every time a candidate service is encountered together with another candidate service in a previous successful composition, the weight of this service is augmented for the current composition (this parameter is tested in lines 7–15). Those two parameters are automatically generated by the execution processor.

The last part in algorithm 4 concerns user feedback regarding previous uses of a service for obtaining a same output. If no information is available for a service regarding one of its outputs, a default neutral value is used during the composition for this service rating. If a quality assessment is found for a service concerning one of the outputs that may be useful for the composition, then this value is used for service selection inside the composition (this parameter is tested in lines 23–31).

This user feedback is not automatically generated. When a service provides the user with a result, the user may validate this result by using the Web interface. When this is done, the quality value associated with this service for the information concerned is augmented.

Input: U = {(I0,V0)...(In, Vn)} ; File F; File H1 ; File H2 ; File H3
Output: File F2 containing the services, with inputs, outputs and associated weights for each output ; Qs the quality assessment for service s
1 S = {}
2 **foreach** *s in F* **do**
3 | S ← S + s
4 **end**
5 EliminateUselessServices(S)
6 **foreach** *s in S* **do**
7 | Qs = 0 **foreach** *workflow W in H1* **do**
8 | | **if** *s in W* **then**
9 | | | **foreach** *service w in W* **do**
10 | | | | **if** *w != s and w ∈ S* **then**
11 | | | | | Qs = Qs + (0.1 * W(H1))
12 | | | | **end**
13 | | | **end**
14 | | **end**
15 | **end**
16 | **foreach** *O in (O0 ... On)* **do**
17 | | **foreach** *(service, output, quality) in H2* **do**
18 | | | **if** *service == s and output == o* **then**
19 | | | | Qs = Qs + (quality * W(H2))
20 | | | **end**
21 | | **end**
22 | **end**
23 | **foreach** *(service, input, value) in H3* **do**
24 | | **if** *service == s* **then**
25 | | | **foreach** *(I, V) in U* **do**
26 | | | | **if** *I == input and value == V* **then**
27 | | | | | Qs = Qs + (0.1 * W(H3))
28 | | | | **end**
29 | | | **end**
30 | | **end**
31 | **end**
32 | Print s, (I0 ... In), (O0 ... On),Qs in F2
33 **end**

Algorithm 4: QoS assessment algorithm

The services that are part of a composition are selected based on individual QoS properties. Approaches comparing entire compositions and choosing a composition over another [12,16] rather than a service over another during the composition phase are not applicable in our case, because of the number of services available. As an example, there are 262 services elected when a composition requires "effective temperature" as an output, and 150 for "heliocentric radial velocity". Exploring every composition possible would lead to 262 * 150 = 39300 possible compositions only for those two outputs. The directed, acyclic graph resulting from the composition algorithm is turned into a directed, acyclic graph with weighted edges. The weight of each edge is the estimated quality for the information provided (the destination node of the edge) given by the service (the source node of the edge). Algorithm 5 shows how useless services are dropped from the composition.

Input: S list of services
Output: S list of services with every useless service
(providing useless outputs for the composition)
removed

```
1  Cflag = 1
2  while Cflag == 1 do
3  │  Cflag = 0
4  │  foreach s in S do
5  │  │  Pflag = 1
6  │  │  foreach O in (O0 … On) do
7  │  │  │  foreach s in S do
8  │  │  │  │  foreach I in (I0 … In) do
9  │  │  │  │  │  if O == I then
10 │  │  │  │  │  │  Pflag = 0
11 │  │  │  │  │  │  Break
12 │  │  │  │  │  end
13 │  │  │  │  end
14 │  │  │  │  if Pflag == 0 then
15 │  │  │  │  │  Break
16 │  │  │  │  end
17 │  │  │  end
18 │  │  │  if Pflag == 0 then
19 │  │  │  │  Break
20 │  │  │  end
21 │  │  end
22 │  │  if Pflag == 1 then
23 │  │  │  S ← S - s
24 │  │  │  Cflag = 1
25 │  │  end
26 │  end
27 end
28 Return S
```

Algorithm 5: deleting useless services algorithm

Input: File F2 from QoS algorithm (Or file F from the composition phase on user specific demand) ; File Fp from composition algorithm containing services parameters (URL if any, and QuerySoftware) ; G = {(I0, U0, F0, V0)...(Ix, Ux, Fx, Vx)} Individual, Unit, Format, Value: information provided by the user
Output: Results from the execution of services

```
1  C={}
2  AI={}
3  foreach s,(I0...In),(O0...On),Qs in F2 do
4  │  C ← C + s,(I0...In),(O0...On)
5  end
6  EliminateUselessServices(C)
7  foreach (I,U,F,V) in G do
8  │  AI ← AI + (I,V)
9  end
10 while C != 0 do
11 │  foreach s,(I0...In),(O0...On) in C do
12 │  │  if (I0...In) in AI then
13 │  │  │  foreach Ix do
14 │  │  │  │  Vx = GetValue(AI,Ix)
15 │  │  │  end
16 │  │  │  url = GetUrl ( Fp, s)
17 │  │  │  querysoft = GetQuerySoftware(Fp,s)
18 │  │  │  Result = RunService(s,url,querysoft, (I0...In), (V0...Vn))
19 │  │  │  Write(H3, V0...Vn,s)
20 │  │  │  AI ←AI + (O0...On)
21 │  │  │  C ← C - s,(I0...In),(O0...On)
22 │  │  │  Write(H1,s)
23 │  │  end
24 │  end
25 end
26 Return Result
```

Algorithm 6: Execution algorithm

3.4. Execution phase

The execution algorithm chooses the best service for each output requested from the composition, based on algorithm 4. Following the topology of the different compositions, this may result in some services becoming useless. This happens when a sub-chain of the composition leads to the use of a given service, and this given service is not the best for any of his outputs. Then, some services in the sub-chain (or every service in the sub-chain, if none of the services in the sub-chain provides another service in the overall composition with the best input possible) may become useless. Algorithm 6 presents the execution algorithm. The Web interface open for users is shown in Fig. 5. The user's only expected action is to specify the information he wants to obtain, according to his own vocabulary and unit habits.

After the execution phase, the user may modify the values of the weights given to the QoS parameters W(H1), W(H2), and W(H3). A new QoS assessment and execution may then be run, without going through the composition phase another time. Running every composition available is also possible. This is done by avoiding the QoS assessment algorithms, and going to the execution phase from the file F resulting from the composition phase. Doing so will query every service elected during the composition, bypassing every QoS assessment and is an "emergency solution" for compositions where very few services contain the required outputs in a certain running context.

4. Experiments

This work does not intent to improve composition timing performances over existing approaches. It aims to present the

difficulties of bringing SWS composition in a domain with specific requirements, and to propose solutions to encompass those difficulties. Nevertheless, several test have been run in order to estimate the performances of the composition system. The results are presented hereafter. Information requested (IR) are the information that have been requested from the system, the inputs being the name of an astrophysical object and a radius around this object. The number of services available (SA) indicates how many services have been part of the composition, either as final information providers or as in-between elements between the inputs and outputs. The time necessary for the composition in seconds (CT) is indicated in Table 1. The time for the "quick orchestration" (QT) in seconds is the time necessary for obtaining all outputs requested by the user; querying only the best service available for each of them. The variations of quick orchestration time are not very large, as the quick orchestration only queries the best service available for each output, plus the services in-between when necessary. On the contrary, the composition discovers every service providing an output plus every service in-between, so the composition time grows with the number of services capable of providing outputs or in-between information.

As a summary, we can see that the composition phase provides workflows combining more than 1000 services in less than 200 s. Such workflows are composed when the user query indicates fuzzy description of very widely provided information. The sorting of services is made on the total amount of services in ASON, which describes 11 135 services in its current version. Every request

Table 1
Experimental results.

IR	SA	CT(s)	QT(s)
magnitude, velocity, temperature, color	1115	195	11
metallicity, radial velocity, rotational velocity, surface gravity	528	123	6
field chart, heliocentric radial velocity, effective temperature, interactive sky view	679	141	16
spectral type, metallicity, temperature, surface gravity	537	126	7
Spectral type, microturbulent velocity, alpha/Fe	61	35	4
luminosity, effective temperature, mass, O/Fe	567	121	5
infrared spectrum, visible spectrum, gamma spectrum, uv spectrum	11	25	5
log axis ratio, apparent total b magnitude, galactic extinction in b band, velocity	335	82	11
magnetic field detection, radial velocity, rotational velocity, surface gravity	526	128	7

presented in the experiments has successfully returned every information expected by the user.

5. Conclusion and future works

This work presents an insight of full automation of semantic Web services composition in the fields of astrophysics. To this end, it simplifies the definition of requirements, ensures the quality of the composition per the specificities of the application context, and automates the composition from the search for services to the production of results.

The main contributions of this paper are the representation of I/O (authorizing specific combinations like the correlated inputs), the ability to compose services Web and non-Web based alike through the use of "QuerySoftware" concept acting as a service driver. The QoS parameters, based on the history of users' previous workflows and quality comments is another contribution.

A system implements the approach and is made available to the astrophysics community at http://cta1.bagn.obs-mip.fr [28]. No technical prerequisites are required to use the system. User feedback is automatically considered for future service compositions. Future work involves broadening the spectrum of available services to very specific branches of astrophysics such as astrochemistry. An extension of this approach to other scientific fields (e.g., geophysics and aerology) is under study.

References

[1] P. Bartalos, M. Bieliková, et al., Automatic dynamic web service composition: A survey and problem formalization, Comput. Inform. 30 (4) (2011) 793–827.
[2] J. Amudhavel, U. Prabhu, P. Inbavalli, N. Moganarangan, V. Ravishankar, R. Baskaran, P. Dhavachelvan, Survey and analysis of web service composition strategies: A state of art performance study, Indian J. Sci. Technol. 9 (11) (2016). URL http://www.indjst.org/index.php/indjst/article/view/89265.
[3] T. Berneers-Lee, J. Hendler, O. Lassila, The semantic web, Sci. Am. 284 (5) (2001) 34–43.
[4] D. Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D.L. McGuinness, E. Sirin, N. Srinivasan, Bringing semantics to web services with OWL-S. World Wide Web 10 (3Martin, D. et al. 2007. Bringing semantics to web services with OWL-S. World Wide Web, 10(3), pp.243–277.), 243–277.
[5] R. Dumitru, K. Uwe, L. Holger, B. deJos, L. Rubén, S. Michael, P. Axel, F. Cristina, B. Cristoph, F. Dieter, Web service modeling ontology, Appl. Ontology 1 (1) (2005) 77–106.
[6] A.L. Lemos, F. Daniel, B. Benatallah, Web service composition: a survey of techniques and tools, ACM Comput. Surv. 48 (3) (2016) 33.
[7] D.C.T. Batista, SHOP2 : An HTN planning system, System 20 (2011) 3–6.
[8] M. Dubernet, B. Antony, Y. Ba, Y. Babikov, K. Bartschat, V. Boudon, B. Braams, H.-K. Chung, F. Daniel, F. Delahaye, G. Zanna, J. Urquijo, M. Dimitrijević, A. Domaracka, M. Doronin, B. Drouin, C. Endres, A. Fazliev, S. Gagarin, I. Gordon, P. Gratier, U. Heiter, C. Hill, D. Jevremović, C. Joblin, A. Kasprzak, E. Krishnakumar, G. Leto, P. Loboda, T. Louge, S. Maclot, B. Marinković, A.

Markwick, T. Marquart, H. Mason, N. Mason, C. Mendoza, A. Mihajlov, T. Millar, N. Moreau, G. Mulas, Y. Pakhomov, P. Palmeri, S. Pancheshnyi, V. Perevalov, N. Piskunov, J. Postler, P. Quinet, E. Quintas-Sánchez, Y. Ralchenko, Y.-J. Rhee, G. Rixon, L. Rothman, E. Roueff, T. Ryabchikova, S. Sahal-Bréchot, P. Scheier, S. Schlemmer, B. Schmitt, E. Stempels, S. Tashkun, J. Tennyson, V. Tyuterev, V. Vujčić, V. Wakelam, N. Walton, O. Zatsarinny, C. Zeippen, C. Zwölf, The virtual atomic and molecular data centre (VAMDC) consortium, J. Phys. B: At. Mol. Opt. Phys. 49 (7) (2016).
[9] S. McIlraith, T. Son, H. Zeng, Semantic web services, IEEE Intell. Syst. 16 (2) (2001) 46–53, Special Issue on the Semantic Web.
[10] C. Pedrinaci, D. Liu, M. Maleshkova, D. Lambert, J. Kopecy, J. Domingue, IServe: A linked services publishing platform, CEUR Workshop Proc. 596 (2010) 71–82.
[11] D. Tosi, S. Morasca, Supporting the semi-automatic semantic annotation of web services: A systematic literature review, Inf. Softw. Technol. 61 (2015) 16–32.
[12] X. Zhao, L.W. Shen, X. Peng, W. Zhao, Finding preferred skyline solutions for SLA-constrained service composition, in: Proceedings - IEEE 20th International Conference on Web Services, ICWS 2013, IEEE, 2013, pp. 195–202.
[13] M.L. Sbodio, D. Martin, C. Moulin, Discovering Semantic Web Services using SPARQL and Intelligent Agents, Web Semant.: Sci. Serv. Agents World Wide Web 8 (4) (2010) 310–328. URL http://www.sciencedirect.com/science/article/pii/S1570826810000533.
[14] J. Puttonen, A. Lobov, J.L.M. Lastra, Semantics-based composition of factory automation processes encapsulated by web services, IEEE Trans. Ind. Inf. 9 (4) (2013) 2349–2359.
[15] Y. Li, Y. Li, T. Hu, Z. Lv, Semantic Web Based on Ontology Method, 2015, pp. 1–4.
[16] P. Rodriguez-Mier, C. Pedrinaci, M. Lama, M. Mucientes, An integrated semantic web service discovery and composition framework, IEEE Trans. Serv. Comput. 9 (4) (2016) 537–550.
[17] S. Bansal, A. Bansal, G. Gupta, M.B. Blake, Generalized semantic Web service composition, Serv. Oriented Comput. Appl. 10 (2) (2014) 111–133.
[18] L. Simon, A. Mallya, A. Bansal, G. Gupta, T.D. Hite, A universal service description language, in: Web Services, IEEE International Conference on, 2005, pp. 823–824. URL http://www.computer.org/plugins/dl/pdf/proceedings/icws/2005/2409/00/24090823.pdf?template=1&loginState=2&userData=Friedrich-Althoff-Consortia+Berlin-Brandenburg+%2528consortium%2529+%257E+Universitaet+Leipzig%253AFriedrich-Althoff-Consortia+Berlin-Brande.
[19] G.a. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.
[20] F. Yong-Yi, S. Yang, M. Shang-Pin, K. Jong-Yih, An overview and classification of services description approaches in automated services composition, IEEE Trans. Serv. Comput. 10 (2) (2017) 176–189.
[21] Q.Z. Sheng, X. Qiao, A.V. Vasilakos, C. Szabo, S. Bourne, X. Xu, Web services composition: A decade's overview, Inform. Sci. 280 (2014) 218–238.
[22] P. Severi, J. Fiadeiro, D. Ekserdjian, Guiding reification in OWL through aggregation, CEUR Workshop Proc. 573 (2010) (2010) 408–419.
[23] P. Dowler, M. Demleitner, M. Taylor, D. Tody, 2014. IVOA Recommendation: DALI: Data Access Layer Interface Version 1.0. CoRR abs/1402.4. URL http://arxiv.org/abs/1402.4750.
[24] T. Louge, M.H. Karray, B. Archimède, J. Knödlseder, Ason: An owl-s based ontology for astrophysical services, Astron. Comput. (2018).
[25] A. Costa, P. Massimino, M. Bandieramonte, U. Becciani, M. Krokos, C. Pistagna, S. Riggi, E. Sciacca, F. Vitello, An innovative science gateway for the cherenkov telescope array, J. Grid Comput. 13 (4) (2015) 547–559.

[26] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, Sov. Phys. Dokl. 10 (8) (1966) 707–710.

[27] W.E. Winkler, The state of record linkage and current research problems, 1999.

[28] T. Louge, M.H. Karray, B. Archimède, J. Knödlseder, CASAS: A tool for composing automatically and semantically astrophysical services, Astron. Comput. 20 (2017) 34–43.

Thierry Louge received his Ph.D. degree in the area of computer science in 2017 under the guidance of Bernard Archimède, Jürgen Knödlseder and Hedi Karray. He is also an IT engineer working in IT for astrophysics since 1999 in a French observatory, the Observatoire Midi-Pyrénées (OMP) in Tarbes, south of France. OMP is part of Université Paul Sabatier (UPS).

Hedi Karrary received his Ph.D. degree in the area of applied informatics from the University of Franche-Comté, France, in 2012. Since then, he has been an Assistant Professor at Haute Alsace University for one year. Since September 2013 he holds an Associate Professor at the National Engineering School of Tarbes of the National Polytechnic Institute of Toulouse, Part of the Federal University of Toulouse, and member of Production Engineering Laboratory. In 2016, he had a visiting researcher position at the State University of New York at Buffalo (UB). Since his stay at UB, Dr. Karray became senior scientist at National Center for Ontological Research heade. The research topics of Dr. Karray center on Ontology-Based Engineering, semantic interoperability and decision support systems.

Bernard Archimède received the Ph.D. degree in Industrial Computer Sciences from the University Bordeaux 1 in 1991. Since this date, he is researcher in the Laboratory Génie de Production (LGP) of the National School of Engineers at Tarbes (ENIT) where he is Professor. His research deals with the distributed software architectures and their applications to the multisite scheduling and the control of complex systems. Since September 2016, he is assistant director of the LGP. He has broad interest on distributed planning architectures, distributed simulation, multi-agents systems and interoperability.

Zakaria Maamar is a Professor in the College of Technological Innovation at Zayed University. His research interests are primarily related to service sciences theories and methods, context-aware computing, and enterprise application interoperability. Dr. Maamar has published several peer-reviewed papers in journals and conferences, is the founder of the International Annual Symposium on Web Services (www.zu.ac.ae/wss), and regularly serves on the program and organizing committees of several international conferences and workshops. In 2010, he received an IBM Faculty Award to carry out research on social Web services. Dr. Maamar graduated for his M.Sc. and Ph.D. in Computer Sciences from Laval University in Canada in 1995 and 1998, respectively.

Dr. Michael Mrissa is full professor at the Universite de Pau et des Pays de l'Adour (UPPA). He received his Ph.D. from the University Claude Bernard Lyon 1 in 2007 and his accreditation to supervise research (HDR) in 2014. His main research interests relate to service-oriented computing, semantic reasoning and the Web of Things. He has published over 60 peer-reviewed papers in conferences and journals. He is involved in the ANR project ASAWoO1.