# From user goals to process-based service compositions: A flexible semantic-based approach

Isabelle Mirbel

# From User Goals to Process-based Service Compositions: a Flexible Semantic-based Approach

Isabelle Mirbel
WIMMICS Research Team
Universite Cote d'Azur, Inria, CNRS, I3S,
Sophia Antipolis, France
Email: isabelle.mirbel@unice.fr

November 23, 2017

**Abstract**

Complex user's needs often require heterogeneous services to be combined together. In this paper, we explore a novel approach to enable flexible and dynamic composition of heterogeneous services for end users who are usually not familiar with technical process models. End-users are only required to model their goals and semantic reasoning is used for the composition itself. The possible service compositions are expressed in BPMN in order to be then processed by a BPMN engine. Since the composition is built on-the-fly, this approach avoids static linking of services and therefore is much more flexible.

## 1 Motivation

Adaptation and flexibility are new challenges enterprise information systems are facing today. To let end-users take advantage of these information systems, their specific needs have to be captured and turned into technical solutions. Indeed, complex user requirements usually require heterogeneous services to be combined together. Such a combination is usually captured into a more or less complex workflow specified in a dedicated manner. In this paper, we explore a novel approach to enable flexible and dynamic composition of heterogeneous services for end-users who are usually not familiar with technical process models. Support is provided to fill the gap between goal statements and technical process models. We propose a goal-driven approach, where end-users state their main goals and our system determines whether the goal can be reached given the set of available services. It also infers what service compositions are necessary to reach it. The possible service compositions are expressed in BPMN (Business

Process Model and Notation) in order to be then processed by a BPMN engine. This approach suits dynamic service environments, since service compositions are created at runtime according to stated user goals and available services.

The rest of the paper is organized as follows. In section 2, we describe our proposal allowing end-users to state their goals as semantically-annotated maps, enriched with context information. We also explain how map can be stored to help for future use. Then, in section 3, we discuss how our prototype supports reasoning capabilities on top of semantically-annotated maps. Related works are presented in section 4. Finally section 5 concludes the paper.

# 2 Modeling Goals with Maps

On the one hand, complex user requirements required heterogeneous services to be combined together. Such a composition is usually captured into a more or less complex workflow specified in a dedicated manner. BPMN is an example of dedicated notation for complex process modeling. On the other hand, end-users are usually not familiar with technical process models such as BPMN. They are more used to goal statement for instance. Moreover, the use of a goal model instead of a business process model language such as BPMN is motivated by the fact that a goal model allows to consider alternative generation of process specification for a specific context [**?**]. Business process modeling languages focus on sequences of activities and synchronization, but do not model the business goals that the processes achieve. On the contrary, goal models allow to specify and select between alternative activities and sequencing.

## 2.1 The Map Model

We rely on the Map model which has been introduced in the nineties in the field of Information System Engineering [**?**] and validated in several other fields among which process modeling [**?**]. This model introduces an intentional level into process modeling. It focuses on what the process is intended to achieve, thus providing the process rationale, i.e. why the process is performed. The Map model allows to specify process models in a flexible way by focusing on the process intentions, and on the various ways to achieve each of these intentions.

A map is a diagram in which nodes are intentions and edges are strategies. The directed nature of this diagram shows which intentions should precede which one another. Therefore, it is not imposed that once an intention is achieved the intention that immediately follows is undertaken. An edge enters a node if its associated strategy can be used to achieve the target intention. Since there can be several edges entering a node, a map is able to gather different ways of achieving an intention.

A map includes two predefined intentions, `Start` and `Stop`, to show the beginning and the end of a process. In the Map model, a particular process step is modeled as a section. A section is defined by a triplet $<$ *source intention*, *strategy*, *target intention* $>$ in which :

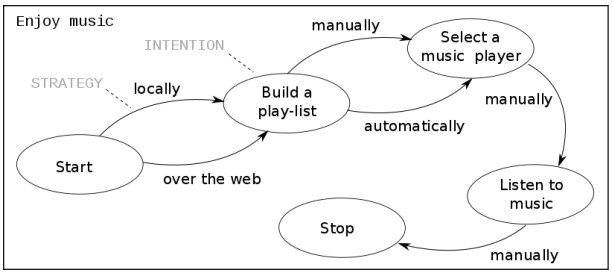


Figure 1: Example of map

- the *source intention* represents a specific situation,
- the *strategy* describes a particular technique used to achieve the target intention,
- the *target intention* represents the goal of the step.

One way to achieve an intention is captured in a map section whereas all sections having the same source and target intentions represent all the different strategies that may be used to achieve this target intention from the situation obtained after the realisation of the source intention. In the same way, there may be several sections with the same source intention but different target ones. These sections show all the intentions that can be reached after the source intention has been achieved.

Figure **??** shows a map example. Note that in the Map model, the section is not represented with a unique symbol but with a set of concepts (source intention, target intention and strategy). Thus, the section concept doesn't appear as such in a map.

As a map is able to gather different ways of achieving an intention, means have been provided in the Map model to guide the reader while selecting a path.

- If more than one target intentions are reachable from a particular source intention, an *ISG (Intention Selection Guideline)* is associated to the source intention to specify how to select the next target intention(s).
- If more than one strategy exists to reach a specific target intention from a particular source intention, a *SSG (Strategy Selection Guideline)* is associated to the pair of source and target intentions to specify how to select the right strategy(ies).
- An *IAG* (Intention Achievement Guideline) may be associated to a section to guide its practical enactment. In our case, an *IAG* is either another map describing more in detail how to fulfill the target intention, or a query to select an appropriate service to achieve the target intention.

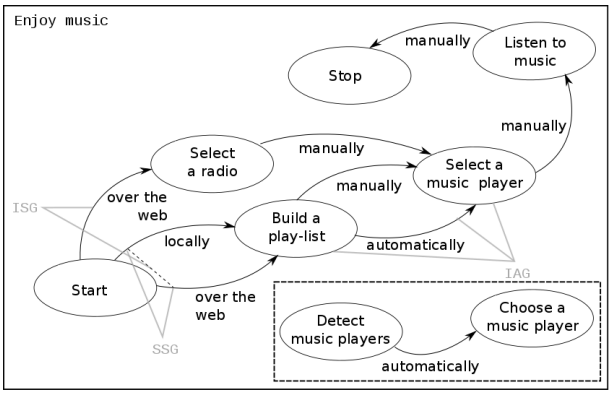


Figure 2: Example of *ISG*, *SSG* and *IAG*

Examples of *ISG*, *SSG* and *IAG* are shown in figure **??**

Indeed, a map is a navigational structure which contains a finite number of paths, each of them prescribing a way to achieve the goal associated with the whole process described by the map. As a result, the Map model allows to model intention-oriented process.

## 2.2 Semantic Annotation of Maps

To further formalize intentions and strategies, we rely on Prat's proposal [**?**], which has already proven to be useful to formalize goals [**?**, **?**]. According to Prat [**?**], an intention is characterized by a verb and some parameters which play specific roles with respect to the verb. Among the parameters, there is the object on which the action described by the verb is processed.

In order to provide capabilities for reasoning on maps, we propose to semantically annotate them. In our previous work, we proposed an Resource Description Framework Schema (RDFS) ontology for maps which gathers the concepts of the Map model [**?**]. The following example shows a map section semantically annotated with the Map Ontology. It corresponds to one of the two first sections of the map presented in figure **??**.

```
@prefix mo: <http://map-onto/> .
_:S1  a mo:Section ;
      mo:hasStrategy _:St1;
      mo:hasSource [a mo:StartIntention] ;
      mo:hasTarget [a mo:Intention ;
                mo:hasVerb "build" ;
              mo:hasObject "play-list"] .
_:St1 a mo:Strategy ;
      rdfs:label "locally" .
```

## 2.3 Enhancing the Map Model for Dynamicity and Flexibility

As previously stated in the paper, our proposal takes place in a dynamic context where services appear and disappear requiring flexibility in the way user's needs are turned into service calls. In the current Map model, guidelines to decide how to refine or implement a map section (i.e how to achieve a goal or a sub-goal) is documented in natural language and thus not allowing an automated processing of the maps. In order to support dynamic selection of services, we improved the Map model by providing a set of predefined operators to specify guidelines and associated semantic annotations to be able to reason on these operators. Moreover, as it has been explained above, the Map model allows to specify different paths (i.e. different means) to achieve a target intention (i.e. a user goal statement). And the workable path is built following the situation at hand. As our aim is to automate map processing, we also improved the Map model to take into account contextual information. We present these two improvements in the following sections.

### 2.3.1 Automating map execution

In the Map model, guidelines are currently provided in natural language. As a consequence, they can only be exploited by human beings. As our aim is to be able to automatically process the maps in order to build a BPMN representation of a process according to the available services, we formalized guidelines writing. Three binary operators are proposed:

- `PAR` which means that two sections must be processed in parallel;
- `OR` which means that at least one section must be processed and that one section can be processed after the other;
- `XOR` which means that one of the two sections must be processed only.

Figure **??** shows an example of automatically processable guidelines in which `I2` identifies the RDF annotation of the `Select a radio` intention, `I3` identifies the RDF annotation of the `Build a play-list` intention, `St1` refers to the strategy presented in the previous annotation example and `St2` identifies the RDF annotation of the strategy labeled `over the web` and linking the `Start` intention to the `Build a play-list` intention. We also enrich the Map Ontology in order to be able to reason on these operators.

### 2.3.2 Modeling Contextual Information

Flexibility in the way user's needs are turned into service calls also requires being aware of the end-user context when selecting the workable path in a map. Therefore, contextual information has to be embedded into maps in order to be exploited at runtime when reusing a map to fulfill a user's need. For this purpose, we propose to introduce guards on strategies. Figure **??** shows the

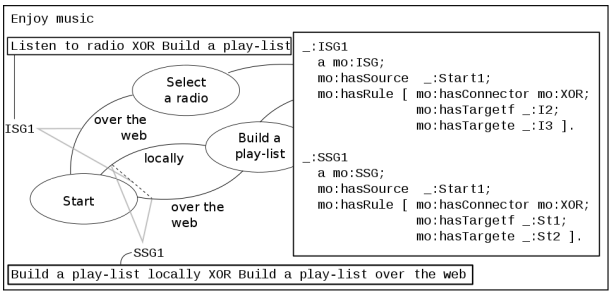


Figure 3: Example of automatically processable guidelines

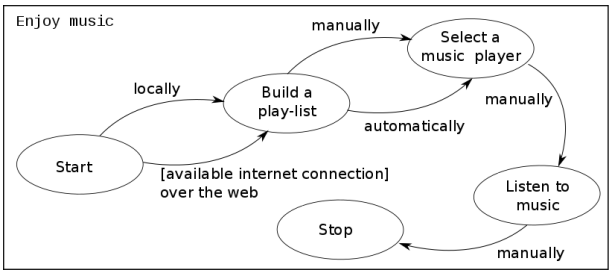


Figure 4: Example of contextual information

example of figure **??** in which a guard has been introduced for strategy `over the web` in order to explain in which context the strategy is meaningful.

As explained previously, when looking for available services with the help of a (set of) maps, sections are turned into more refined maps or queries to retrieve (at least one) appropriate service(s) to make the target intention achievable; And guards are replaced by dedicated queries too in order to find the appropriate service(s) able to check at runtime if the context makes the strategy workable or not. So, we enriched the Map model with a *Guard Achievement Guideline (GAG)* concept to associate queries to guards.

In the enriched example of figure **??**, at runtime, if no service is available to check the internet connection or if no internet connection is available, the `over the web` strategy will be considered as not possible. As the *SSG* associated to the `Start` intention indicates an `XOR` operator between the two possible strategies, the target intention will still be reachable if a service fulfilling the `locally` strategy is found. With a `PAR` operator the target intention would have been unreachable and the user's need (i.e. the full map) not achievable.

Comparison operators (such as for instance $<$, $>$ or $=$) may be used to define a guard. In this case, several queries must be associated to the guard to make it processable at run time. For instance, if a strategy is meaningful only when the outside temperature is higher than the inside temperature, then the guard will

be expressed as follows: `outside-temperature > inside-temperature`; and two queries will be associated to the guard: one to select a service to get the outside temperature and one to get the inside temperature.

Thanks to this improvement of the Map model with regards to contextual information, we introduced much more flexibility in the on-the-fly composition building process.

### 2.4 Map Sharing

In the context of an enterprise information system, map annotations are stored in a dedicated knowledge base (i.e. a SPARQL endpoint) where they will be shared. Later, they will be retrieved when needed to help an end-user to achieve his/her main goal.

The knowledge base may gather several *IAG* associated with a specific section in order to specify different means to reach the target intention following the same strategy. As a given section may appear in different maps, by storing all maps in the same SPARQL endpoint, the *IAG* defined for a section in one map are suitable for all the maps in which this section appears. Thus, maps stored in the same knowledge base enrich one another. In other words, our approach supports modularity.

More generic *IAG* may also be provided, outside of the context of any map. For this, the section to which the *IAG* is associated may be described either by its target intention (no source intention and no strategy are specified) or by its strategy and its target intention (without a source intention). In this way, the *IAG* is susceptible to be suitable for a higher number of maps.

## 3 Reasoning on Maps

Given the statement of a user goal, our prototype determines whether it can be reached given the set of available services, and also infers which service composition is necessary to reach it. In other words, our system provides traceability capabilities explaining how a high level goal may be achieved.

In our proposal, goals are specified as intentions belonging to map sections; and each map section includes an intention achievement guideline. An *IAG* is either another map describing more in detail the steps required to reach the target intention, or a query specifying how to look for a service to achieve the target intention. As map are semantically-annotated in RDF relying on our RDFS Map ontology, queries are expressed in SPARQL.

So, given the statement of a user goal, determining whether the goal can be reached consists in looking at the closest intention among the ones already stored in our knowledge base; and then transforming it into a BPMN diagram whose services are resulting from SPARQL queries on the SPARQL endpoint publishing the currently available services. For this transformation, we rely on STTL [**?**](SPARQL-based TransformaTion Language), a generic transformation rule language for RDF based on SPARQL.

In the remainder of this section, we first present the STTL language. Then, we discuss the rules we provided to determine whether the end-user's main objective is achievable. Finally we look at the rules we proposed to infer the service composition.

## 3.1 STTL : A SPARQL-based Transformation Language for RDF

STTL is a lightweight syntactic extension of SPARQL enabling the writing of transformation rules [**?**]. It relies on two extensions of SPARQL: an additional TEMPLATE query form to express transformation rules and extension functions to recursively call the processing of a template into one another.

A TEMPLATE query is made of a standard WHERE clause and a TEMPLATE clause. The WHERE clause is the condition part of a rule, specifying the nodes in the RDF graph to be selected for the transformation. The TEMPLATE clause specifies the output of the transformation for the RDF statements matching the condition. An example of template aiming at translating an *SSG* with a *PAR* operator (WHERE clause) into a BPMN parallel gateway XML definition (TEMPLATE clause) is shown in appendix.

In STTL, several SPARQL extension functions have been defined to process a transformation. They allow to call the transformer again on a focus node or to call another template by its name. Hierarchical processing of a set of templates is done using one of these functions in the TEMPLATE clause. It returns the result of the application of other templates to the focus nodes. The result is concatenated in the TEMPLATE clause. In appendix, we provide an example of template which aims at detecting the type of the operator used in an *SSG* (WHERE clause) to call the right template to transform it into BPMN.

An RDF transformer is a set of transformation rules processed by a generic transformation rule engine. We developed two RDF transformers:

- The *dependency transformer* (i) transforms a goal statement into a sequence of SPARQL queries, (ii) executes the selected queries against the RDF data set describing the available services and (iii) finally determines whether the user goal can be satisfied or not. The output of this first RDF transformer is a *Dependency Model* which will be described more in detail in the next subsection.

- The *behavioral transformer* changes the obtained sequence of SPARQL query results, according to the map specification, into a BPMN representation. The output of this second transformer is a *Behavioral Model*, as it will be detailed in the remaining of this section.

## 3.2 Dependency Model

Our first RDF transformer is decomposed into two sets of templates.

The first set of templates aims at selecting the RDF annotations describing the maps and the SPARQL queries to select suitable services. Starting from

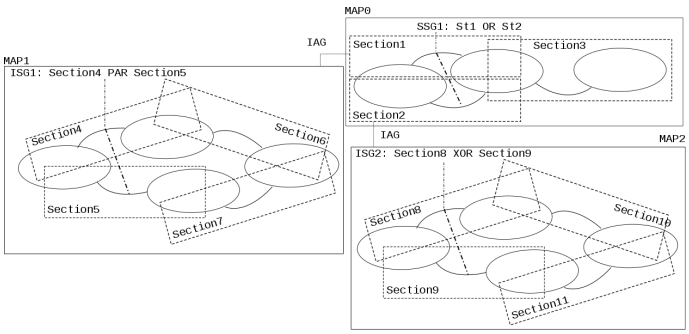


Figure 5: Planning of goal Achievement

the section whose target intention corresponds to the user goal, the templates we provided are recursively executed to select more refined maps or SPARQL queries until all sections have been associated either with a map or with a SPARQL query. Figure **??** shows an example of map (`MAP0`) in which `Section1` is refined into `MAP1` and `Section2` is refined into `MAP2`. Queries are also associated to each section of `MAP1` and `MAP2` as well as to `Section2` and `Section3` (for the sake of readability, queries are not shown in the figure). Indeed, because a query and a map are associated to `Section2`, this section may be supported either by a composition of services whose aggregation is documented in `MAP2` or by a single more complex service.

In this RDF transformer, we distinguish *map templates* to select a section associated with a more refined map from *service templates* to select a section associated with a query to look for available services. Assuming services are described in a RDF compatible way (OWL-S for instance), we provide a set of *service templates* dedicated to this RDF compatible annotation language. However, our approach is generic enough to be used with other kinds of RDF annotations. In this case, it will only require another set of *service templates* to be provided. As a starting point, in our current implementation, we rely on the *Profile* part of the OWL-S specification to select services. We plan to exploit *input* and *output* specifications as well in the future.

Starting from the RDF annotations describing the available services and the selected map sections, our second set of templates checks if the set of available services fully covers the end-user goal. Indeed, a complex end-user goal is likely to be refined into a sequence of sub-goals for each of which at least one available service has to be found. Sub-goals are combined with different kinds of operators (PAR, OR, XOR) which have to be exploited to decide if the end-user goal is achievable or not. We provided several templates to take into account for each map: the sequence of sections, the section decomposition and the navigation rules. Each template returns a true or false answer allowing finally to determine if the end-user goal is achievable or not. Figure **??** illustrates the second step performed by the transformer.

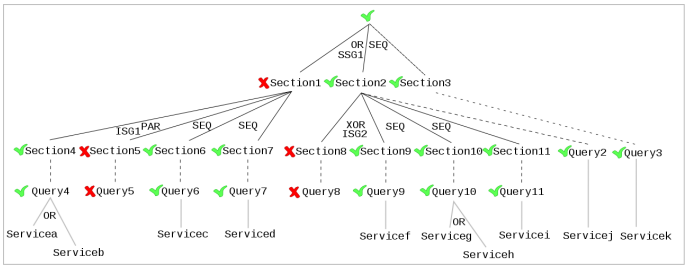


Figure 6: Checking goal achievability

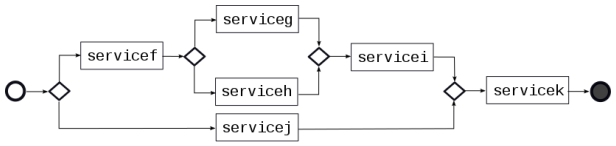


Figure 7: Generated BPMN diagram

In figure **??** example, `Query5` and `Query8` failed, resulting in `Section5` and `Section8` failure. As `Section5` is involved in an *ISG* including a *PAR* operator, `Section1` is not achievable. On the contrary, `Section8` is part of an *SSG* including an *XOR* operator; and as `Section9` is achievable, `Section2` is reachable too. In some cases (`Query4` for instance), several services have been returned. The services will be inserted in the BPMN diagram as alternatives, as it will be shown in the following.

## 3.3 Behavioral Model

Starting from the retrieved services and the achievable map sections, our second transformer aims at building a BPMN 2.0 representation of the service composition required to satisfy the end-user goal. Indeed, map content is transformed into BPMN tags in order to build an XML file executable by a BPMN engine. For this, we provided templates to build the process model part of the XML file (while the process diagrams part is built with default values).

Figure **??** shows the BPMN diagram obtained from the services and map sections selected in figure **??**.

Figure **??** shows the architecture of our prototype embedding the *dependency transformer* and the *behavioral transformer*. Both transformers rely on embedded transformation rules. In our implementation, we provide a set of *service templates* dedicated to OWL-S annotation language. However, our approach is generic enough to be used with other kinds of RDF annotations. In this case, the set of *service templates* has to be replaced by a set of templates dedicated to the targeted RDF annotation language. To use our prototype in a particu-

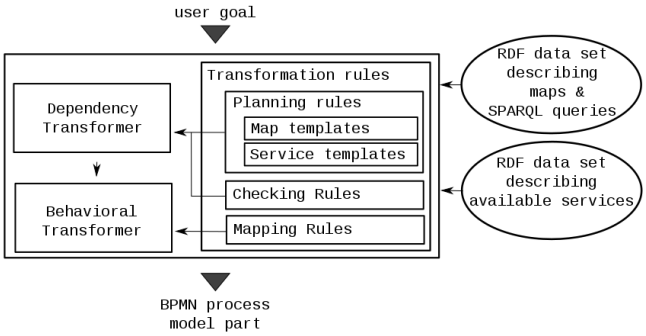


Figure 8: Prototype architecture

lar application domain, it is required to provided the RDF data set describing maps and SPARQL queries as well as the RDF data set describing the available services, in addition to the underlying ontologies.

The quality of the mapping between intention and services is the major limitation of our approach. To address this issue in our future work, we plan to rely on existing work on matching algorithms [**?**, **?**].

# 4 Related Works

In this paper, we suggest a method to semi-automatically specify a service composition by generating a BPMN specification from end-users goals. Indeed we provided templates to build the process model part of the XML file. The process diagrams part is built with default values and has to be manually finalized.

End-user goal modeling has been studied in the domain of Web service retrieval [**?**, **?**, **?**, **?**]. In these proposals, different models are provided to specify goals without addressing the problem of how to capture them. On the contrary, our aim is to provide means to assist final users in querying the Web Service registry to find Web Services to operationalise a business process. The GODO approach [**?**], for instance, addresses this issue by proposing models and tools to capture end-user goals with the help of an ontology or in natural language. In these proposals, support is provided to associate goals to Web services without addressing the problem of how to assist end-users in turning their high level goals into Web services. As in the approach of Kaabi [**?**], we propose an incremental process to refine users' requirements in order to specify the features required for the Web Services under retrieval. Our approach distinguishes itself from the one of Kaabi [**?**] by the fact that we rely on semantic Web models and techniques to enrich the end-user goal specification, in order to provide reasoning and explanation capabilities.

With regards to approaches dealing with ontology-based service discovery [**?**],

and more precisely OWL-S based approaches (as we are relying on OWL-S with regards to Web Service descriptions), capability matching algorithms [**?**] exploiting service profile descriptions have been proposed. Matchmaking algorithms [**?**] comparing state transformations described in the query to the ones provided in the descriptions have also been proposed. All these algorithms mainly exploit features of subsumption relationships. Ranking mechanisms have also been provided [**?**]. Our approach, also ontology-based, distinguishes itself from these works by the fact that our focus is on providing means to assist final users in authoring queries (more than rendering them). In other words, we are interested in the upstream process of deriving queries from final users requirements. Moreover, our concern is also on how to annotate such queries in order to support their capitalisation and sharing among a community of end-users.

Regarding works dealing with goal models and business processes, Gillain et al. [**?**] and Ghose et al. [**?**] propose approaches to derive business processes from user's goals. These two proposals differ from ours by the fact that they are not based on semantic web languages and techniques. Moreover, the approach of Gillain et al. only targets BPEL processes. By relying on semantic web languages and technologies our approach supports reasoning based on the ontologies we rely on. We can for instance reason on more generic or specific verbs and objects used to specify goals. Moreover, semantic web annotation of goal allows one to query the map repository to look for know-how about how to decompose high level goals. The automation of the map model has also been studied from a model-driven approach point of view [**?**, **?**] in order to derive a software architecture for a process enactment engine.

Finally, goal-driven composition has also been applied in the context of the Web of Things [**?**]. To capture end-user goal, the authors rely on a JavaScript-based visual programming tool they extended with components that represent the different predicates useful to describe a smart environment. Indeed, in this approach, goals are states to be reached and no abstraction levels are supported. In Eslami et al. [**?**], an approach dedicated to homecare services is discussed. Goals are decomposed into tasks associated with services. It differs from our approach in which goal decompositions are reused from one map to another in order to enrich each others.

# 5 Conclusion

In this paper we presented a goal-driven approach which allows end-users to state their goals as semantically-annotated maps, enriched with context information. Annotated map diagrams are stored in a shared knowledge base and reused at runtime when an end-user is looking for services to achieve a specific goal.

To find out whether the system can provide a composition that achieves the user goal requires a way of determining what functionality the individual services provide and how to use them. Therefore we rely on semantic annotation of web services. Using semantics within service descriptions represents a lightweight

approach to support new services in an evolving way. To be able to create service compositions, our system must have access to a means of discovering the individual services that are available in the current environment, as well as the semantic metadata they embed. Any SPARQL endpoint that allows to browse or search for semantic service descriptions can be used. In our current work, we rely on CORESE/KGRAM, a semantic web factory (triple store and SPARQL endpoint) implementing RDF, RDFS, SPARQL 1.1 Query & Update [**?**].

Reasoning capabilities are provided to determine whether the user's main objective is achievable and to infer the service composition. Our reasoner relies on a SPARQL-based Transformation Language for RDF [**?**] (i) to turn end-user goal statement into web service descriptions and (ii) to transform goal decomposition guidance rules into BPMN diagrams. Because the composition is built on-the-fly by inferring a path to the end-user goal, this method avoids static linking of services and therefore is much more flexible.

By adopting a goal-driven approach, the complexity of the composition process as a whole can be reduced: users are only required to model their goals and semantic reasoning is used for the composition itself. Because the composition is built on-the-fly by inferring a path to the end-user goal, this method avoids static linking of services and therefore is much more flexible.

Our future works will first be dedicated to end user validation on real case studies. This validation will also allow us to address scalability issues.

In addition, we also plan to better exploit the OWL-S specification. Currently, we rely on the *Profile* part of the OWL-S specification to select services. In the future, we plan to rely on *input* and *output* specifications as well. Moreover, we plan to rely on existing work on matching algorithms [**?**, **?**] to improve the matching between intention and services.

We also plan to look at other semantic annotations, like RESTDesc which has been extended for Web services in smart environments [**?**], in order to also support service calls.

To further explore semantic web languages and technologies and to take into account customization of end-user environments, we will also investigate how to take advantage of tailored ontologies to capture end-user preferences, and thus to derive service compositions that are better adapted to individuals. We also plan to provide means to reuse parts of existing business processes.

## Acknowledgements

We thank Reda Zarhbouch for his contribution in this work, especially for producing the different templates composing the dependency and behavioral transformers.

We also thank Zeina Azmeh for comments that greatly improved the manuscript.

## References


[1] B. Benatallah, M.S. Hacid, C. Rey, F. Toumani, *Request rewriting-based Web Service discovery*. In: Fensel D, Sycara K, Mylopoulos J (eds) Proceedings of the international Semantic Web conference (ISWC 2003), Sanibel Island, FL, October 2003. Lecture notes in computer science, vol. 2870. Springer, Berlin Heidelberg NewYork, pp. 242-257.

[2] L.O. Bonino da Silva Santos, L. Ferreira Pires, M.J. van Sinderen, A Goal-Based Framework for Dynamic Service Discovery and Composition. International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing, July 2008, Porto, Portugal. pp. 67-78.

[3] O. Corby, C. Faron-Zucker. STTL: A SPARQL-based Transformation Language for RDF. 11th International Conference on Web Information Systems and Technologies, May 2015, Lisbon, Portugal. 2015.

[4] O. Corby, C. Faron-Zucker, and I. Mirbel. Implementation of Intention-Driven Search Processes by SPARQL Queries. (Poster) International Conference on Enterprise Information Systems, Milan, Italy, May, 2009.

[5] O. Corby, A. Gaignard, C. Faron-Zucker, J. Montagnat. KGRAM Versatile Inference and Query Engine for the Web of Linked Data. Web Intelligence 2012: 121-128.

[6] R. Deneckere, E. Kornyshova, C. Rolland. Enhancing the guidance of the intentional model "MAP": Graph theory application. Research Challenges in Information Science (RCIS), April 2009, Fes, Morocco. pp.13-22.

[7] M.Z. Eslami, A. Zarghami, B. Sapkota, M. van Sinderen. Service Tailoring: Towards Personalized Homecare Services. ACT4SOC 2010: 109-121.

[8] A.K. Ghose, N.C. Narendra, K. Ponnalagu, A. Panda, A. Gohad. Goal-Driven Business Process Derivation. ICSOC 2011: 467-476.

[9] J. Gillain, S. Faulkner, I. Jureta, M. Snoeck. Using goals and customizable services to improve adaptability of process-based service compositions. RCIS 2013: 1-9.

[10] J.M. Gomez, M. Rico, F. Garcia-Sanchez, GODO: Goal Oriented Discovery for Semantic Web Services. 5th International Semantic Web Conference. 2006.

[11] R.S. Kaabi, *Une approche méthodologique pour la modélisation intentionnelle de services et leur opérationnalisation*. PhD Thesis, Université Paris I Sorbonne. February 2007.

[12] L. Li and I. Horrocks, *A Software Framework for Matchmaking Based on Semantic Web Technology*. International Journal of Electronic Commerce. Volume 8 , Issue 4, Pages 39-60, 2004.

[13] S.D. Mallouli, S. Assar, C. Souveyet, Process behavior meta-modeling for the derivation of enactment engines. RCIS 2014.

[14] S.D. Mallouli, Meta-modlisation du Comportement d'un Modele de Processus : Une Demarche de Construction d'un Moteur d'Execution. PhD thesis, University Paris I Pantheon - Sorbonne, July 2014.

[15] D. Martin M. burnstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D.L. McGuinness, E. Sirin and N. Srinivasan, *Bringing Semantics to Web Services with OWL-S*. World Wide Web (2007). 10:243-277.

[16] S. Mayer. Interacting with the Web of Things. PhD thesis, Eidgenossische Technische Hochschule ETH Zurich, Nr. 22203, 2014.

[17] I. Mirbel, P. Crescenzo. Improving collaboration in the neuroscientist community. International Journal of Web Portals, 3(1), 2011, pp. 33-49.

[18] M. Paolucci, T. Kawamura, T.R. Payne and K. Sycara, *Semantic Matching of Web Services Capabilities*. In First Int. Semantic Web Conference, Sardinia, Italy, June 2002.

[19] N. Prat. Reutilisation de la trace par apprentissage dans un environnement pour l'ingenierie des processus. PhD thesis, Universite Paris I - Sorbonne.

[20] C. Rolland. Conceptual Modelling in Information Systems Engineering, chapter Capturing System Intentionality with Maps. Springer-Verlag. 2007.

[21] M. Stollberg, B. Norton A Refined Goal Model for Semantic Web Services. Second International Conference on Internet and Web Applications and Services (ICIW'07), 2007.

[22] M. Vukovic, P. Robinson GoalMorph: Partial Goal Satisfaction for Flexible Service Composition. International Conference on Next Generation Web Services Practices. 2005.

[23] K. Zhang, Q. Li, Q. Sui A Goal-driven Approach of Service Composition for Pervasive Computing. 1st International Symposium on Pervasive Computing and Applications, pp. 593-598, August 2006.

## Examples of STTL templates

In the following template, the WHERE clause matches the RDF statement and enables to select all the triples and to bind them to variable ?xx in the TEMPLATE clause. The TEMPLATE clause specifies the result that must be generated using the solution sequence of the WHERE clause. Variables in the TEMPLATE clause are replaced by their value.

```
prefix mo: <http://myorg.com/voc/mo#> .

template st:bssg_and(?src, ?trg) {

   """targetElement="_BPMNShape_Parallel
   Gateway_"""xsd:string(?xx)"""><di:wa
   ypoint xsi:type="dc:Point" x="250" y=
   "144.0"/><di:waypoint xsi:type="dc:Po
   int" x="250" y="144.0"/></bpmndi:BPMN
   Edge> """ """ <bpmndi:BPMNShape id="_
   BPMNShape_ParallelGateway_"""xsd:stri
   ng(?xx)"""bpmnElement="""?xx"""><dc:
   Bounds height="50.0" width="50.0" x="8
   87.0" y="185.0"/></bpmndi:BPMNShape><b
   pmndi:BPMNEdge id="BPMNEdge_SequenceFl
   ow_b102" bpmnElement="SequenceFlow_b10
   2" sourceElement="_BPMNShape_ParallelG
   ateway_"""xsd:string(?xx) """\ """"

} where {

  ?x a mo:SSG.
  ?x mo:hasSource ?src.
  ?x mo:hasTarget ?trg
  ?x mo:hasRule ?t.
  ?t mo:hasConnector mo:PAR.
  ?t mo:hasTargete ?te
  ?t mo:hasTargetf ?tf
  bind(strafter( ?x , ":") as ?xx)

}
```

The following example of template aims at detecting the type of the operator used in a *SSG* (WHERE clause) to call the right template (`st:ssg_and` or `st:ssg_rest`) to transform it into BPMN (TEMPLATE clause).

```
prefix mo: <http://myorg.com/voc/mo#> .

template st:ssg_test(?src, ?trg, ?s) {

if( ?ct = mo:PAR ,  st:call-template(st:
      ssg_and, ?src, ?trg, ?s), st:call-te
      mplate(st:ssg_rest, ?src ,?trg, ?s))

} where {

?x a mo:SSG.
?x mo:hasSource ?src.
?x mo:hasTarget ?trg
?x mo:hasRule ?t.
?t mo:hasConnector ?ct.
?t mo:hasTargete ?te
?t mo:hasTargetf ?tf

}
```