

# A PRACTICAL ONTOLOGY-DRIVEN WORKFLOW COMPOSITION FRAMEWORK

Huy Pham, Deborah Stacey and Rozita Dara

*School of Computer Science, University of Guelph, Guelph, Canada*

**Keywords:** Workflow composition, Ontology-driven planning framework, Planning ontology, Ontology-driven workflow composition, Workflow composition.

**Abstract:** Existing planning-based approaches to ontology-driven workflow composition (ODWC) integrate planning into their frameworks and ontologies in ways that are either less reusable or ineffective. A more modular and reusable design is possible, but a successful application of this design requires addressing some important practicality issues. In this paper, we provide a comprehensive survey of existing approaches to intelligent ontology-driven workflow composition, discuss the practicality challenges associated with building ontology-driven and planning-based workflow composition systems, and present a reusable and practical ontology-driven planning framework that could be used for such purposes.

## 1 INTRODUCTION

There are many application domains where a major activity is the creation or arranging of sequences of actions or events. This activity of temporal planning is often referred to as workflow composition. Business processes are often conceptualized as a workflow and thus tools for helping with workflow composition will find a wide and appreciative audience.

From a computational perspective, workflow composition is the task of coming up with an appropriate sequence of (computational) tasks that accomplish a given set of objectives. A simple example is the process that a data analyst may go through when faced with a data mining task. The analyst may employ several data cleaning programs to make sure that the data is consistent and in the proper format. They will then use these data sets as input to a series of different analysis programs and then send the output of these programs to a visualization application to display the results in a format accessible for business analysts.

Recent work in this area has taken advantage of ontologies to drive this composition process. But many existing approaches to planning-based Ontology-Driven Workflow Composition (ODWC) are not totally adequate for building effective real-world workflow composition systems. Our analysis leads us to believe that there are two major gaps that need to be addressed in going from a domain expert with a problem to solve and a system that allows that

expert to translate their domain problem into a computational/computing problem that can generate a design for an appropriate workflow composition.

In this paper, we will provide a comprehensive survey of existing approaches to intelligent ontology-driven workflow composition and classify them into four major approaches. We will then discuss the challenges associated with building ontology-driven and planning-based workflow composition systems and identify features that are desirable in an effective ODWC framework. We then present our reusable and practical ontology-driven planning framework that incorporates all these features to provide the ODWC system designer with a practical, effective and friendly means of building ODWC systems. This framework also addresses the important theoretical challenges that are inherent to the task of doing goal-based reasoning in ontology-driven applications. We hope that our framework functions as a practical way to address this challenge using existing and mature technologies.

## 2 ONTOLOGY-DRIVEN WORKFLOW COMPOSITION

In this section, we start out by reviewing some of the most notable works in ODWC and make some observations about their advantages and limitations. We

then place ourselves in the shoes of an ODWC system designer and explore the list of features that are desirable and important.

## 2.1 Existing Works

From an AI perspective, workflow composition is a form of sequential decision making (Russell and Norvig, 2002), and as such can be approached using various existing knowledge-based planning techniques (Brachman and Levesque, 2004). The main drawback of such an approach however, is that it requires the knowledge bases – the collection of domain knowledge needed to support the workflow composition process – to be implicitly encoded in a language that is specific to the planning framework employed, and therefore less reusable to other applications (Gibson and Stevens, 2009).

In light of this reusability concern, the Semantic Web community has been promoting the use of *ontologies* (Horrocks, 2008) as its key knowledge representation mechanism. This mechanism provides a standardized and well-supported facility that allows domain knowledge to be described in a generic, application-independent way that it can be easily understood, shared and reused by many different applications, including future applications for which the knowledge was not originally intended.

Because of this important advantage that ontologies offer, there has been a lot of interest in representing the knowledge bases of workflow composition systems using ontologies instead of framework-specific and application-specific languages, and consequently, a large number of works have been reported on ontology-driven workflow and system composition. The remaining part of this section provides a review of some of the most notable works, as well as some general observations about the common limitations found in these works.

Most existing works in ODWC use the same overall approach: they *wrap* the various workflow activities into logical units, called compositional units (CU), adorn each CU with an appropriate set of attributes (*e.g.* name, function, input data format, etc.), organize them into some hierarchical structure and serve them up – via an ontology – as a set of building blocks from which a workflow can be constructed. Where they differ is the mechanism through which each of these frameworks produce their workflows. In this respect, existing works on ODWC can be roughly divided into four different approaches.

### 2.1.1 Interactive Composition

In this approach, workflows are composed in an interactive fashion using manual inputs from the user. The software composition framework described in (Hlomani and Stacey, 2009) is one such example. This framework aims to provide the foundations needed to enable non-technical domain experts to rapidly compose and modify their own software systems from individual building blocks that represent the various computing algorithms. More specifically, the framework uses an ontology to formally describe the various computing algorithms, and offers them as a set of building blocks from which the user can visually compose her computational system by selecting and connecting individual algorithmic components together. One of the main limitations of this approach is that while intelligent assistance is provided, the user is still required to come up with the design.

### 2.1.2 Template-based Composition

In the second approach, workflows are selected from a library of pre-built workflow templates and suggested to the user as a guideline for composing her system. This approach is represented by the data mining workflow composition framework described in (Morik and Scholz, 2003). The primary rationale behind this approach is that successful and effective workflows, once built, tested and tuned by experts, could, and should, be shared and reused as guiding templates for non-expert users in similar applications. The main advantage of this approach is that such a framework can act as a facility for easily capturing, sharing and reusing effective data mining workflows. The main limitation is that user assistance can only be provided for situations in which a similar and appropriate template can be found in the library.

### 2.1.3 Planning-based Composition via Direct Translation

In this approach, workflow composition is treated as a planning problem and a planning mechanism is employed to generate workflows. Some of the most notable approaches are described in (Bernstein et al., 2005), (Žáková et al., 2008) and (Diamantini et al., 2009). The main drawback of this approach is that typically all planning-related knowledge is implicitly encoded into the primary knowledge base (*i.e.* the ontologies used to describe the compositional units). Because planning-related knowledge is application and problem specific, embedding them into the main ontologies makes these ontologies application-

and purpose-dependent and, as a consequence, less reusable for future applications.

#### 2.1.4 Planning-based Composition via a Planning Ontology

This approach is represented by the works described in (Gil et al., 2000) and (Rajpathak and Motta, 2004). These papers promote the use of a dedicated and generic Planning Ontology to describe planning-related knowledge.

In (Gil et al., 2000), the authors describe an ontology that contains language constructs for describing, among other things:

- Objectives, subobjectives, and objectives decomposition
- Tasks (actions), task constraints, and *tasks-accomplish-objectives* types of relationships
- Decision points, ordering of tasks, temporal constraints

The strong point about this ontology is that the list of language constructs it provides is comprehensive and powerful enough to describes complex plans. The main limitation of this work is that because its main intent was to describe *plans* as opposed to *planning problems* (the first is a solution of the second), the ontology lacks the necessary constructs (pre-conditions, effects, world's dynamics, cost/reward, etc.) needed to drive a planner, and therefore is inadequate for the purpose of building a planning-based ODWC system.

## 2.2 Desirable Features of an ODWC Framework

In this section, we place ourselves in the shoes of the ODWC system designer and explore the list of features that are important or desirable for ODWC systems. We will do this by trying to understand the types of support the system designer would need in order to build an effective and practical planning-based ODWC system.

On the architectural side, the system designer needs a framework that allows him to create simple and reusable knowledge bases, as well as a framework that can be employed to build WC systems in different domains without extensive modifications or custom coding. On the representation side, he will need a comprehensive set of language constructs for describing real-world workflow composition problems. On the reasoning side, he will need a practical mechanism to do goal-based reasoning in his system.

### 2.2.1 Clear Separation between Planning Knowledge and other Domain Knowledge

This approach offers at least 3 important advantages over previous approaches to planning-based ODWC.

1. *Increased reusability of domain knowledge:* Because the domain and objectives ontologies are independent from the application that uses them, they are much more reusable for other applications.
2. *Easier to build descriptive domain ontologies:* Because the ontology designers can focus on describing the domains without having to worry about how those descriptions will be used later, high quality and descriptive domain ontologies become easier to build.
3. *Increased reusability and sharability of planning knowledge and strategies:* Because the syntax and language constructs would allow planning problems to be declaratively described in a manner that is completely independent from the implementation of the workflow composition engine, the Planning Ontology has the potential to act as a common abstraction language in which planning problems and strategies can be encoded in a sharable and reusable way.

### 2.2.2 Planning Knowledge should be Captured using an Ontology instead of Directly in the Underlying Planning Formalism

The use of a planning ontology would allow the workflow composition problem, and the logic for how to solve it, to be modelled independently from the underlying planning paradigm. This separation is a very important feature because it would allow the underlying translation framework to be reused for building other ODWC systems without requiring a major modification to its logic. All that needs to be modified instead is the ontological problem description.

### 2.2.3 Should provide a Comprehensive Set of Language Constructs to Effectively describe Real-world Workflow Composition Problems

As discussed earlier, automated workflow composition, from a computational point of view, can be considered a planning problem (Russell and Norvig, 2002) and, as such, can be modelled or described in pretty much the same way as a planning problem. A typical workflow composition problem can be adequately modelled using the following types of descriptions:

- Workflow Actions
- Workflow Actions Pre-Conditions
- Workflow Actions Effects (aka, Post-Conditions)
- Initial State
- Goal State

Once the above model has been built, it can be fed into the workflow composition engine (WCE), which searches through the many possible combinations of action sequences and comes up with an appropriate workflow that solves the problem.

In order to be able to build effective ODWC systems that can handle complex, real-world problems, the system designer will need a comprehensive set of language constructs that can effectively describe, at a minimum, all the knowledge types above. While existing ontology-driven planning frameworks do provide some of these constructs, they still lack some very important ones, including those needed to describe the world’s states (e.g. initial state, goal state), state transitions, as well as those needed for advanced planning techniques such as heuristics.

**2.2.4 Should provide a Practical Mechanism for doing Goal-based Reasoning in ODWC Applications**

From a knowledge representation and reasoning perspective, workflow composition boils down to the task of proving the following entailment:

$$KB \models \exists w \text{Valid}(w) \wedge \text{AchieveGoal}(w)$$

where  $w = [a_1, a_2, \dots, a_N]$  is the workflow the user is interested in, and  $KB$  is the knowledge base representing the workflow composition problem.

While several other knowledge-based formalisms provide native support for this kind of reasoning, it is not easily attainable in an ontology-driven application. Based closely on Description Logics (DL), ontologies were intended as a formalism for describing (object-oriented) worlds and is not well-suited for goal-based reasoning tasks such as this.

First Order Logic (FOL) is the most expressive language compared to DL and Horn Logic (HL) but is undecidable. DL and HL are less expressive yet decidable subsets of FOL. These two languages can be thought of as two different ways of attaining decidability via limiting the scope of the language. Due to the difference in the ways their scopes are limited, each of these two languages are well-suited for different purposes – DL is well-suited for describing (object-oriented) worlds, while HL is well-suited for goal-based reasoning and hence problem solving.

While several approaches have been proposed to integrate goal-based reasoning (e.g. Rules) into Description Logics (c.f. the Semantic Web Rule Language (SWRL) (Horrocks et al., 2004), and Description Logic Programs (DLP) (Grosz et al., 2003)), their success to date is still somewhat limited. As discussed in (Hitzler and Parsia, 2009), this is a non-trivial task. The major challenge is that modifying or extending the underlying language (DL in this case) could lead to undecidability. SWRL, for example, is undecidable and, as a consequence, has no native reasoner. Even more importantly, introducing a new extension to a language often entails some difficult and time-consuming tasks. First, all the theoretical results would need to be carefully mapped out and secondly, the toolset will be need to be updated/augmented. Thirdly, getting users to accept the new language is also a highly non-trivial task. It usually takes a long time for a language or formalism to attain the needed critical mass for widespread adoption.

Due to these inherent difficulties, hybrid approaches are often considered when goal-based reasoning is needed within an ontology-driven application. In section 3 below, we propose a hybrid (i.e. translational) solution for integrating goal-based reasoning into ODWC systems.

**3 A PRACTICAL ODWC FRAMEWORK**

**3.1 The Overall Architecture**

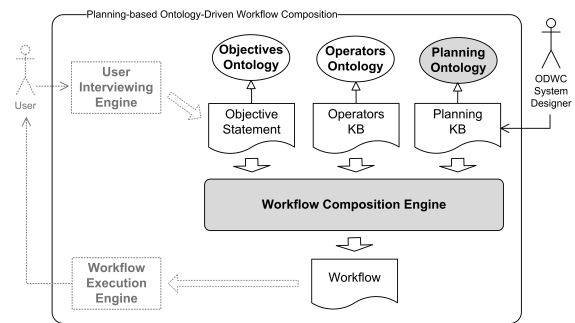


Figure 1: A generic architecture for planning-based workflow composition systems. The planning ontology and workflow composition engine constitute the proposed framework and are reusable across domains and applications.

In our proposed architecture, as illustrated in Figure 1, a workflow composition system would utilize three different ontologies to come up with its workflows.



First, an *Objectives Ontology* is used to describe all the workflow objectives that the system is expected to accept as its inputs. In a software project management ODWC system, for instance, this objective ontology is used to describe objectives such as “Produce a workflow that would minimize bug count” or “Produce a workflow that would minimize time to market”, etc.

Similarly, an *Operators Ontology* will be used to describe all the compositional units from which the workflow can be composed. In a data mining ODWC system, for example, the Operator KB will contain the descriptions for all the different data mining and preprocessing algorithms.

Finally, all planning-related knowledge needed to drive the planner and build the workflow from the individual compositional units is captured in the planning KB. This KB is described by a domain and reusable Planning Ontology. As discussed in section 2.2.3 above, this ontology provides all the necessary language constructs needed to describe workflow composition problems. A simplified version of this ontology is shown in Figure 2.

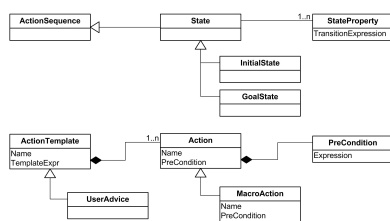


Figure 2: A simplified view of the Planning Ontology showing basic planning constructs such as states and actions.

It is worth noting that this architecture incorporates two of the desirable features identified in section 2.2 above. Namely, it maintains a clear separation between planning-related knowledge and other domain knowledge allowing the domain knowledge to stay purpose-independent and hence more reusable for future applications. It also allows the planning logics to be described and modelled in an ontology instead of directly in the underlying planning formalism. As discussed earlier, this design feature allows the framework to stay application-independent and can be reused to build different workflow composition systems for different domains.

### 3.2 Integrating Goal-based Reasoning into Ontology-driven Applications

As discussed in section 2.2.4 above, ontologies and DLs do not provide native support for goal-based reasoning. We have taken a translational approach in

which the planning problem, once it has been described by the ODWC system designer using the Planning Ontology, is translated into an equivalent executable program in HL and then executed using a Prolog-based planning engine.

#### 3.2.1 Model Closure

One of the standard cautions one has to take when integrating goal-based reasoning with ontological modelling is the open-world vs closed-world assumptions conflict. Goal-based program’s KBs are closed-world models (a fact can be assumed to be false if it has not been stated otherwise), while ontological models are open-world models (a fact cannot be assumed false unless it was explicitly asserted). Currently however, this is not a practical issue in our situation – the ODWC designer just has to picture a closed world in his mind when describing his planning problem using the Planning Ontology. To help make it mentally explicit for the designer however, we provide ontological constructs that he can optionally insert into the KB to logically (*i.e.* mentally) “close” the model down. The statement “No other workflow actions are available”, for example, when inserted into the KB, has the effect of finalizing the list of already asserted workflow actions and hence provides a mental closure to the model.

#### 3.2.2 Ensuring Translatability

Also, as with other translational approaches, one of the main theoretical questions that is of importance to our proposed approach is that of translatability – how does one ensure that the planning problem descriptions created by the ODWC system designer are always translatable into an executable planning program in HL? To answer this question, a few observations are in order.

First, a well-defined ontology can be thought of as a form of language – the list of concepts it provides constitutes the vocabulary of the language, while the roles it defines dictates the ways in which the vocabulary can be combined together to form statements. Secondly, by carefully controlling the list of concepts and roles in the ontology, we can restrict or control the types of statements one can express using the ontology.

With these observations in mind, one can see that by being very selective and careful with the language constructs in the Planning Ontology, we can ensure that all possible workflow composition problem descriptions are translatable to executable planning programs in HL. This, in fact, is the main intuition behind our approach. Figure 3 provide a visual illustration

for this intuition.

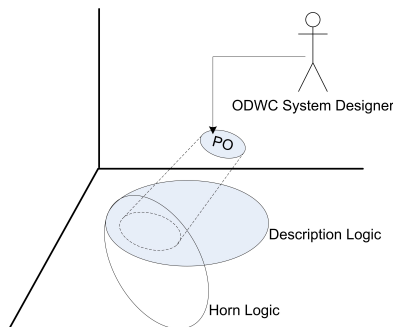


Figure 3: The primary intuition behind our proposed approach: the Planning Ontology acts as a restrainer that helps ensure the description of the planning problem always falls into a sub area of Description Logics that is translatable to an executable program in Horn Logic

### 3.2.3 Practical Benefits

By taking a translational approach, as opposed to a language modification or extension approach, our framework is able to make use of an existing and mature theoretical framework (HL) and technologies (Prolog) to provide seamless goal-based reasoning capability in ontology-driven applications.

Another advantage offered by our framework is that it allows the ODWC system designer to continue to think and work in the ontological modelling environment that he or she is already comfortable with, without having to learn a new language extension or a new planning formalism.

## 3.3 Addressing Effectiveness Issues

### 3.3.1 Incorporating the System Designer's Heuristic Advice

Real-world workflow composition problems often result in highly complex planning problems that are well beyond the capabilities of even the most advanced automated planners. Fortunately, in almost all cases, the ODWC framework designer will have some valuable procedural insights on how to take advantage of the problem's structure or how to best go about solving the problem. An effective and practical planning-based ODWC framework, therefore, must provide convenient facilities and mechanisms for the designer to provide these insights to his system. In our framework, this can be done via a mechanism called *partial programming* (Reiter, 2001). The idea is that, instead of relying solely on the planner to generate workflows from scratch, the workflow compo-

sition system would start from a partial template that the system designer has provided. Because this template contains all the heuristic advice from the designer, computing a workflow from this template will be much faster and efficient than computing one from scratch.

Using our framework, the system designer creates these partial programs (*i.e.* templates) using the language constructs provided by the Planning Ontology. We describe some of these language constructs below:

- **Action Template.** This concept is used to represent a subworkflow, and is the equivalence of a procedure in a programming language. An action template can contain workflow actions (*i.e.* compositional units) or other action templates. The constituents of the template can be arranged in a simple sequential order or in complex procedural orders such as loops (while, foreach) or branch (if..then..else).
- **ChoicePoint.** This concept is used to represent a nondeterministic choice between two or more ActionTemplates (*i.e.* subworkflows). The designer would use this construct to convey to his workflow composition system that, instead of having to consider all possibilities, it can narrow its choices to just the subset of subworkflows specified in the choice point. The more insight the designer has about the workflow composition problem, the more choice points he will put in the template and the less work the system has to do.
- **ArgChoicePoint.** Each workflow action might take one or more input arguments and an ArgChoicePoint can be used to narrow down the possible argument values the system has to consider.

### 3.3.2 Hierarchical Composition

Our Planning Ontology also provides language constructs for dealing with hierarchical workflow composition. In particular, the ODWC system designer can use the MacroAction concept to annotate (or *wrap*) a subworkflow into a compositional unit, with its own pre-conditions and effects. The workflow composition system can then use these macro units, alongside with other regular units in a seamless manner, to quickly compute high-level workflows satisfying the objective. Once such a workflow has been successfully found, the macro units can be iteratively fleshed out into concrete subworkflows.

## 4 SUMMARY

In this paper, we reviewed the overall landscape of ontology-driven workflow composition, and described the four major approaches to which existing works can be classified. We have also identified and explained a list of features that are desirable for an effective ODWC framework and proposed a practical framework that incorporates all these features to provide the ODWC system designer with a practical, effective and friendly means of building ODWC systems.

We have also provided a discussion on how important theoretical challenges that are inherent to the task of doing goal-based reasoning in ontology-driven applications can be practically addressed using existing and mature technologies.

As for future works, we are working to provide a formal proof on the translatability of workflow composition problem descriptions. Additionally, we are also working to incorporate the notions of planning cost/reward and concurrency into our planning framework to allow it to produce non-linear workflows that not only accomplish the give objective, but also accomplish it in an optimal way. Also, each workflow action can have non-deterministic effects, and we are also looking to incorporating non-determinism into our framework.

A longer version of this paper including a case study can be found at [ontology.socs.uoguelph.ca](http://ontology.socs.uoguelph.ca).

## REFERENCES

- Bernstein, A., Provost, F., and Hill, S. (2005). Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering*, 17:503–518.
- Brachman, R. and Levesque, H. (2004). *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Diamantini, C., Potena, D., and Storti, E. (2009). Ontology-driven kdd process composition. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII, IDA '09*, pages 285–296, Berlin, Heidelberg, Springer-Verlag.
- Gibson, A. and Stevens, R. (2009). Introduction to ontologies. In Popescu, M. and Xu, D., editors, *Data Mining Applications Using Ontologies in Biomedicine*. Artech Hous.
- Gil, Y., Gil, A., and Blythe, J. (2000). PLANET: A Shareable and Reusable Ontology for Representing Plans. In *Proceedings of the AAAI Workshop on Representational Issues*.
- Grosov, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: combining logic programs with description logic. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pages 48–57, New York, NY, USA. ACM.
- Hitzler, P. and Parsia, B. (2009). Ontologies and rules. In Bernus, P., Blazewics, J., Schmidt, G., Shaw, M., Staab, S., and Studer, R., editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 111–132. Springer Berlin Heidelberg.
- Hlomani, H. and Stacey, D. (2009). An ontology-driven approach to software system composition. In *Proceedings of KEOD 2009*.
- Horrocks, I. (2008). Ontologies and the Semantic Web. *Commun. ACM*, 51(12):58–67.
- Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosov, B., and Dean, M. (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Technical report, World Wide Web Consortium.
- Morik, K. and Scholz, M. (2003). The miningmart approach to knowledge discovery in databases. In In Ning Zhong and Jiming Liu, editors, *Intelligent Technologies for Information Analysis*, pages 47–65. Springer.
- Rajpathak, D. and Motta, E. (2004). An Ontological Formalization of the Planning Task. In *Proceedings International Conference on Formal Ontologies in Information Systems (FOIS'04)*, Torino, Italy.
- Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, Massachusetts, MA, illustrated edition edition.
- Russell, S. J. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall.
- Žáková, M., Křemen, P., Železný, F., and Lavrač, N. (2008). Using ontological reasoning and planning for data mining workflow composition. In *SoKD: ECML/PKDD 2008 Workshop on Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery*.