APPROACH FOR VERIFYING WORKFLOW VALIDITY

Yuan Lin, Thérèse Libourel, Isabelle Mougenot
LIRMM, 161 rue Ada - Espace DEV, 500 rue JF Breton - University of Montpellier 2, Montpellier, France

Runtong Zhang, Rongqian Ni
Institute of Information Management, Beijing Jiaotong University, #3 Shangyuancun, Haidian District, Beijing, China

Keywords: Scientific workflow, Workflow validation, Process composition, Resource hierarchy.

Abstract: This article presents the solution adopted for tackling the problem of incompatibility inherent in process compositions during a workflow’s construction. The proposed approach is based on a context of pre-constructed resource hierarchies (data and processes) and consists of finding possible composition “paths” between processes within GRSYN and GRSEM resource graphs constructed from the context. We explain the stage of constructing the context from a simple formal description of resources. The stage for resolving the incompatibility is then covered in detail. We briefly present the implemented prototype before highlighting future avenues of research.

1 INTRODUCTION

Scientific domains dealing with topics such as biodiversity, ecology, and agronomy require the drawing up of experimentation plans using various resources (data and processes). These resources, while available in ever-increasing quantities, remain, for the most part, expensive – and thus their reuse becomes almost a necessity.

To design these complex experiments, scientists often need to locate suitable resources and then to organize or reorganize them. In addition, each experimentation plan deserves to be saved so that it can be re-executed several times, either in various different configurations or with diverse test data. In such a context, the use of a scientific workflow proves to be an invaluable help. Several dedicated software applications for this purpose now exist and research in the field is relatively advanced. A first study (Libourel et al., 2010) presented the concept of the workflow environment. Our approach aims to help the user:

- design experimentation plans (in as abstract a manner as possible),
- better organize resources (data and processes) which will be elements in the concretization of these plans,
- capitalize on the existing by constructing new processes from previously devised plans.

This article develops our research advances in terms of resource organization and semi-automatic verification of validity of workflows designed within a prototype.

Section 2 lists the problem to be addressed, the work context and the definitions we will rely on for our validation approach. Section 3 presents a state of the art on process composition. Section 4 explains the validation process in terms of algorithms. Section 5 presents the prototype. Section 6 concludes our proposal by listing planned perspectives.

2 PROBLEM AND CONTEXT

Referring again to the idea (Libourel et al., 2010) that experimentation requires a stage of abstract planning followed by a concretisation stage in which the user selects the most suitable data and processes, we aim to address the basic problem of the validation of the concrete experimental chain.

In figure 1, the user designs, in a biological context, an experiment in which he aligns sequences followed by a tree reconstruction based on the alignment results. To do so, he uses two concrete processes, Blastx and PhyML.1

1In the adopted graphic formalism, “abstract” and “con-
APPENDAGE FOR VERIFYING WORKFLOW VALIDITY

Figure 1: Problem.

Validation of a workflow consists of verifying the compatibility of each of its composition based on the concept of work context (which we will define in greater detail in section 4).

A composition between two processes corresponds to a link between the input parameter and output parameter of these processes.

The verification of a composition's compatibility ensures its later executability. Various approaches can be considered, for example, that of analyzing process signatures or one based on the analysis of differences between communication protocols or on methods governing exchanges between heterogeneous and distributed systems, etc.

As far as we are concerned, we will mainly focus on the verification of signatures of two linked processes. The signature of a process encompasses, in our opinion, two important aspects:

- The syntactic aspect, which defines the data formats used by each parameter.
- The semantic aspect, which determines a process's functionality. This not only concerns the process's name (which has to make sense) but also the significance of each input and output parameter.

The verification of a composition's compatibility will therefore relate to these two levels: the syntactic and the semantic. However, before presenting the verification algorithms for workflow validation, we first survey the existing approaches from which we have drawn inspiration.

3 PROCESS COMPOSITION: A STATE OF THE ART

Our survey consists of two parts: one concerning various representative projects, the other discussing different existing research efforts concentrating on the problem of compatibility.

creté” processes are represented by rectangles, and input and output parameters by circles. Data flow is represented by arrows.

3.1 Workflow Environments

All the environments we list use graphical interfaces. These permit scientists to construct experimental plans using distinct formalisms. Nevertheless, all of these environments are located at a level that we call “concrete”.

Kepler (Ludäscher et al., 2006; Altintas et al., 2006) is a complete scientific workflow environment based on the Ptolemy II platform of the University of Berkeley. In this environment, actors correspond to different possible processes and operations, and they are equipped with ports representing their input/output parameters. The compositions between processes are made interactively by scientists by linking actor ports with channels. The control and orchestration of the workflow model is the responsibility of directors. Necessary adaptations are made via intermediary programs (senders and receivers), thus ensuring compatibility of data transferred via a channel.

Taverna (Hull et al., 2006; Oinn et al., 2006) is a workflow project created by the e-Grid team in England and used mainly in the biological domains. Processes in this environment are essentially web services (which can be supplemented by local libraries, manuscript scripts, etc.). During process composition, the user manually couples input/output parameters of web services or invokes shim services, specific adaptors designed earlier from couplings made for already constructed and tested experiments.

NetBeans is a general-purpose IDE environment. One of its modules allows the construction, via the use of the BPEL (Business Process Execution Language) (Andrews et al., 2003), language of workflows by the composition of web services. A thorough knowledge of the BPEL standard is however required. The composition is done by manual coupling or transformation between XML elements of exchanged messages. These coupling rules are then translated with the help of the XSLT language (eXtensible Stylesheet Language Transformations) (Kay, 2007).

Weka (Cunningham and Denize, 1993) is an application from the machine learning and data mining domains, created by the University of Waikato, New Zealand. It includes one component, Weka KnowledgeFlow, which allows chaining of processes relating to data mining experimentation. The general model of KnowledgeFlow follows the sequence Selecting data → Filtering → Classifying → Evaluating → Visualizing. Thanks to Weka’s graphical interface, scientists can interactively concretize their experiments, and choose pre-existing converters to ensure their workflows’ compatibility. The environment
is based on data categories and algorithms relating to various processes constructed beforehand.

### 3.2 Existing Approaches

The approaches we list below are essentially those relating to the semantics of processes. They originate from the domain of artificial intelligence.

**Ontological Approach.** The ontological approach assumes the pre-existence of domain ontologies, constructed beforehand for the resources (data/processes), by using standards such as OWL (Group, 2004). During the design of the workflow, the user locates resources and composes processes guided by these ontologies. For example, in the METEOR-S project of the University of Georgia, USA, the workflow system controls the compatibility of the chaining of web services by using the SAWSDL extension (Joel Farrell, 2007) for establishing relationships between WDSL descriptions (Christensen et al., 2001) of these web services with the concepts of an OWL ontology. In (Liu et al., 2007), web-service messages are expressed in the form of RDF graphs (W3C, 2004). Compatibility is verified by pairing between these graphs and the ontology concepts.

**Planner Approach.** In the field of artificial intelligence, planners are used when, to attain a fixed objective, an action plan is considered. In a workflow context, planner algorithms can help find all possible process compositions so as to obtain, given a description of an initial state, the final desired state. The authors of the article (Beauche and Poizat, 2008) use two specific structures: CSS (Capacity Semantic Structure), which represents the workflow in the form of a tree, with nodes being either abstract processes or control operators (sequence, choice or parallelism); and DSS (Data Semantic Structure), which represents the structure of data allowable for each process. The planner calculates all the possibilities of constructing the workflow using the services chosen by the users. Several plans can be proposed that take the adaptation of the concerned DSS's into account. The plan selected by the user is transformed into YAWL orchestrators (van der Aalst and ter Hofstede, 2005). A prototype based on this approach has been implemented (GraphAdaptor). The article (Klusch and Gerber, 2005) uses a set of web service descriptions in OWL-S (Martin et al., 2004) and an associated OWL ontology. They are converted into the PDDL language (Planning Domain Definition Language). The Xplan planner can, using these translations, calculate various possible plans that will allow the predetermined objective to be attained. Similarly, the article (Sirin et al., 2004) shows how to use the SHOP2 planner (Nau et al., 2003) for arriving at plans of web services compositions (described in OWL-S). The article (Julien Bourdon and Fiorino, 2007) uses a multi-agent architecture for the planning of web services using an interaction between agents (services) for achieving the predetermined goal. The article (Claro et al., 2008) uses the SPOC system planner (CLARO, 2006) for determining and putting in sequence the web services discovered in the initial localization stage. It offers an optimization of the planning process based on the user’s profile.

**Other Approaches.** The articles (Limthanmaphon and Zhang, 2003) use case-based reasoning approaches. The process chain is created after learning from analogous cases (composition) and adaptation to the target context.

### 3.3 Summary

The work surveyed focuses, for the most part, on the composition of web services. Ontological descriptions prove to be essential in detecting semantic incompatibilities. The adaptations require transformations between incompatible message structures. The planner approach is not necessarily natural and can prove complex and demanding for users who are not experts in informatics. Therefore, we have retained essentially the “ontological” approaches but we wish to provide an environment in which process chains can reach beyond web services to invoke libraries and specific processes.

## 4 Our Approach

Given the problem stated in section 2, we thus propose an approach based on the analysis of a workflow’s compositions and guided by the concept of work context (cf. sub-section 4.1). We define different types of composition compatibilities in sub-section 4.2. From this categorization, we identify three compatibility situations, which we then put through a semi-automatic repair algorithm (cf. sub-section 4.3).

### 4.1 Work Context

The verification of a workflow’s validity consists of verifying the compatibility of each of its compositions in terms of the work context. This work context consists of three major organizations or arrangements of resource descriptions, namely:
• Organization of human resources, for managing
  the description of the users of the platform and
  their various roles and associated access rights.
• Organization of data, for managing data cate-
  gories, concrete data, and the various associated
  data formats.
• Organization of processes, for managing the de-
  scription of process categories and concrete pro-
  cesses.

The concept of Converter introduced in the figure
2 refers to the concept of a specific process to con-
vert between different formats of data belonging to
the same data category3.

To construct this environment, a simple formal-
ism, designed for resource descriptions has been pro-
posed and formalized by using XML schemas4. Fig-
ure 3 depicts the schematic structures of data- and
process-description categories, of data formats, and of
the data and processes themselves. Properties of these
resources are listed within parentheses. The different
levels of the structure represent its sub-elements, with
the numbers before each element indicating its cardin-
ality. The relationships between these descriptions
are guaranteed by the references stored in the various
elements, for example, the DataCategoryRef element
in the Input element of a TaskDescription points to the
data category that this parameter uses. The arrows labelled ref.DC
and ref.FO connect these parameters (input and output) to the associated data
categories and formats. And, finally, the signature of the Blastx process can be represented as
Blastx(NucleicSeqs:txt):(ProteinSeqs:txt).

4.2 Verification of Composition
Compatibility in our Context

Verifying the conformity of a workflow’s composition
before execution consists of detecting and correct-
ing the incompatibilities in each of its compositions.
More specifically, it is a matter of verifying the com-
patibility of the two ends (parameters) of each link.
The formal description of signatures proposed for the
processes allows us to define the concepts of syntac-
tic and semantic compatibilities. Let two processes
T1 and T2 be described by the following signatures:

Figure 2: Work context.

Figure 3: Schematic representation of the simple formalism
designed for description of resources.

as ovals), ordered on the basis of the generalization/specialization relationship. The description of
cong crete resources (data or processes) are then associ-
ated with their category. A set of data formats (bare, ttx, Fasta, Tgf, Newick) is also presented.

To take both the syntactic and semantic aspects
into account, we propose the following formal de-
scription for the signature of every concrete process: Name (list of Param J) : (list of Param O), with

• Name representing the name of the concrete pro-
  cess or operation.
• Param J and Param O which represent, respec-
  tively, one of the input or output parameters p. p
  is of the form (dc:fo), with dc and fo relating to
  the data category and format used.

Graphically, both aspects, syntactic and semantic,
of each signature are represented by dashed arrows
which connect concrete processes with their data cate-
gories and formats. For example, the graphical de-
scription of the Blastx process shows that this process
has one input parameter and one output parameter.
The arrows labelled ref.DC and ref.FO connect these
parameters (input and output) to the associated data
categories and formats. And, finally, the signature of the Blastx process can be represented as
Blastx(NucleicSeqs:txt):(ProteinSeqs:txt).
In our work context, only the descriptions are saved.

From these two definitions, we develop our proposed approach for resolving the incompatibilities.

4.3 Repairing an Incompatible Composition

Of the three situations we have arrived at, the latter two require additional adaptations before moving on to the execution stage.

The general procedure that is used to verify the validity of a workflow’s composition corresponds to the following algorithm 1. Repair($p_1$, $p_2$). This procedure can trigger two types of adaptations: semantic adaptation to overcome semantic incompatibility and syntactic adaptation to do the same with syntactic incompatibility.

To illustrate our approach, a sample dataset has been created. It consists of definitions of 10 data categories and 4 integrated data formats, as well as of 14 descriptions of processes, of which 3 are converters.
Algorithm 1: Repair(p1, p2).

Input: Parameter p1, Parameter p2
begin

Situation =
DetermineCompatibleSituation(p1, p2);
if Situation == 1 then
  ok;
end
else if Situation == 2 then
  SyntacticAdaptation(p1, p2);
  //select one of the proposed solutions
  UpdateComposition();
end
else if Situation == 3 then
  SemanticAdaptation(p1, p2);
  //select one of the proposed solutions
  UpdateComposition();
  for All sub-compositions px→py added between p1→p2 do
    SyntacticAdaptation(px, py);
    // select one of the proposed solutions
    UpdateComposition();
  end
end

TD111, TD121 and TD131. Their signatures are:

TD1(DC1:FO1) : (DC2:FO2),
TD2(DC2:FO1) : (DC3:FO2, DC4:FO1),
TD3(DC3:FO3) : (DC5:FO1),
TD4(DC3:FO2) : (DC6:FO4),
TD5(DC4:FO3) : (DC8:FO2),
TD6(DC5:FO1, DC6:FO2) : (DC7:FO3),
TD7(DC1:FO1) : (DC3:FO2, DC4:FO4),
TD8(DC1:FO1) : (DC1:FO3),
TD9(DC8:FO2) : (DC7:FO4, DC9:FO2),
TD10(DC4:FO1) : (DC4:FO2, DC7:FO3),
TD11(DC7:FO4) : (DC3:FO2),
TD12(DC10:FO2) : (DC7:FO4),
TD111(DC2:FO2) : (DC2:FO3),
TD121(DC4:FO1) : (DC4:FO3),
TD131(DC2:FO3) : (DC2:FO1)

Taking the composition between T1 and T11,
(DC2:FO2) → (DC7:FO4) (cf. fig.5), we see that it corresponds to Situation 3. To validate this composition, we have to find a solution to, first, ensure semantic compatibility, then, as a second step, ensure syntactic compatibility.

These two successive adaptations will require the definition and construction of two types of resource graphs (GRSEM and GRSYN), constructed from the work context.

4.3.1 Semantic Adaptation

For a semantically incompatible composition, the proposed solution consists of finding processes or process compositions which permit the conversion of the source data category into that of the destination. To achieve this first goal, we construct the resource graph (GRSEM).

GRSEM is an oriented graph \( GRSEM = (N, A) \), with:

- A set of nodes \( N = N_P \cup N_{DC} \), with \( N_P \) being the set of process description nodes and \( N_{DC} \) being the set of data category nodes.
- A set of arcs \( A \). If an arc \( a=(n1, n2) \in A \), then \( n1 \in N_P \land n2 \in N_{DC} \lor n1 \in N_{DC} \land n2 \in N_P \).

Two types of arcs are present in the GRSEM, \( A = A_R \cup A_S \):

1. \( A_R \) is the set of reference arcs going to the data categories used by a process parameter. If \( a_r=(n1, n2) \in A_R \), then \( n1 \in N_P \land n2 \in N_{DC} \lor n1 \in N_{DC} \land n2 \in N_P \).
2. \( A_S \) is a set of specialization arcs between data categories. If \( a_s=(n1, n2) \in A_S \), then \( n1 \in N_{DC} \land n2 \in N_{DC} \lor (n1 \text{ represents a direct subcategory of that represented by } n2) \).

The GRSEM of figure 6 was generated from the sample dataset: circular nodes represent data categories, rectangular ones correspond to process descriptions. The reference and specialization arcs are then added between the nodes.

Semantic adaptation can be considered as a path-finding problem between two data category nodes in the GRSEM resource graph.

A recursive algorithm is used. It takes as input the nodes of the two data categories concerned and generates all the possible paths between them in the GRSEM. Each path found includes a set of intermediary nodes and represents a potential semantic adaptation (sequence of intermediary processes). For the composition (DC2:FO2) → (DC7:FO4), and the constructed GRSEM graph, we obtain the following potential adaptations:

\[ \text{From } n1 \text{ to } n2. \]
To undertake this stage, a second, specific resource graph (GRSYN) is constructed using converters⁶.

GRSYN is an oriented graph \( \text{GRSYN} = (N, A) \), with:

- a set of nodes \( N = N_{\text{Comb}} \cup N_{\text{Convert}} \), with \( N_{\text{Comb}} \)
- a set of arcs \( A \). An arc \( a = (n_1, n_2) \in A \) implies \( (n_1 \in N_{\text{Convert}} \land n_2 \in N_{\text{Comb}}) \lor (n_1 \in N_{\text{Comb}} \land n_2 \in N_{\text{Convert}}) \). This set corresponds to the reference links between a converter node and a combined node.

The GRSYN generated using the sample dataset is shown in figure 8.

As is the case for the semantic adaptation, the syntactic adaptation can be considered as a path-finding problem in the GRSYN. Let us consider again the composition between \( T1 \) and \( T11 \): after the first stage of semantic adaptation, we have obtained a new model (cf. fig.7) which is semantically compatible for all its compositions. Only the syntactic compatibility of each “tmpCategoryLink” needs to be verified. Considering the link between \( T1 \) and \( T2 \), the two connected parameters are (DC2:FO2) and (DC2:FO1). Therefore, a syntactic adaptation has to be found between \( FO2 \) and \( FO1 \). A single itinerary was found in our GRSYN: \( \text{DC2} \rightarrow \text{FO1} \rightarrow \text{DC2} \rightarrow \text{TD111} \rightarrow \text{DC2} \rightarrow \text{TD131} \rightarrow \text{DC2} \rightarrow \text{FO1} \). If we retain this solution, the two converters \( \text{TD111} \) are \( \text{TD131} \) are substituted for the “tmpCategoryLink” link between \( T1 \) and \( T2 \). In the same way, we can also establish syntactic adaptations for the composition \( (\text{DC4:FO1}) \rightarrow (\text{DC4:FO3}) \) between \( \text{TD2} \) and \( \text{TD5} \). The final updating of the instantiated workflow (cf. fig.9) corresponds to the replacing of the “tmpCategoryLink” links by the concerned converter(s).

⁶Note that to us, as previously defined, a converter is a specific process which converts data between different formats of the same data category. We thus assume that these converters exist.
According to this approach, the example in figure 1 will require only a syntactical adaptation of the composition \((\text{ProteinSeq:txt}) \rightarrow (\text{ProteinSeq:Fasta})\) which can be achieved using a converter between the \textit{txt} and \textit{Fasta} formats.

### 5 PROTOTYPE

The formal approach was tested via a prototype implemented in Java. This prototype consists of five main modules:

1. **Resource centre**: component responsible for managing resources, itself consisting of two sub-components:
   - (a) **Resource manager** which offers a graphical editor to help enter resource descriptions (these descriptions are then stored locally in XML files).
   - (b) **Search engine** which accommodates requests to search for resources necessary to construct concrete workflows.

2, 3. **Workflow editor** for editing abstract and instantiated workflows. This is a graphical editor which allows workflow models to be constructed. The simple workflow language that we proposed in the article (Lin et al., 2009) is used.

4. **Validation module** is the component that verifies and validates an instantiated model. It provides adaptation solutions to overcome the incompatibility situations encountered.

5. **Learning module**, as yet un-implemented, should allow the enriching of the work context using analyses of models already constructed.

Figure 10 shows the prototype’s functioning in a schematic form. The user first creates an abstract model of the desired experimentation plan. He then proceeds to its instantiation by using the search engine which provides him with the description of concrete resources. The instantiated workflow is then analysed by the validation module before execution takes place.

A demonstration of the prototype is online at http://www.lirmm.fr/lin/project.

### 6 CONCLUSIONS AND PERSPECTIVES

The incompatibility problem discussed in this article is one of the major issues in process composition. The approach we have presented proposes the data flow checking based on the work context, i.e., on a set of pre-constructed resource hierarchies. The algorithms for constructing different types of resource graphs (GRSEM and GRSYN) and for validating a concrete workflow’s compositions are operational in a working prototype.

Planned future research will explore:

- Extension of resource descriptions:
  - Use of formalisms such as richer WSDL or OWL-S for improving resource descriptions.
  - The formalism currently proposed is simple.
The semantic aspect of resource descriptions could be thus complemented. The construction of the work context could benefit from the use of ontologies originating from the target experimental domains.

- The semantic level of process is currently only covered by the name and the parameters’ data categories. They could be extended by using terminological relationships (synonymy, etc.), as well as by adding complementary information to the descriptions relating to the process’s behaviour (state machine, for example).

- The development of the learning module. It could, on the basis of analyses of constructed models, lead to the enriching of the resource centre and the work context (trace analysis, model statistics, etc.).

- The connection of the validated workflow to an execution engine.

Other approaches like type or composition contract checking (Comerio et al., 2009, Milanovic, 2005), behaviour checking based on the Petri-net (Kiepuszewski et al., 2003; Hamadi and Benatallah, 2003) have also been found in lecture. These research results will be taken into account in our future works.

REFERENCES


