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]

[1] In the adopted graphic formalism, "abstract" and "con-

Lin Y., Libourel T., Mougenot I., Zhang R. and Ni R..
APPROACH FOR VERIFYING WORKFLOW VALIDITY.
DOI: 10.5220/0003498900660075
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (ICEIS-2011), pages 66-75
ISBN: 978-989-8425-55-3

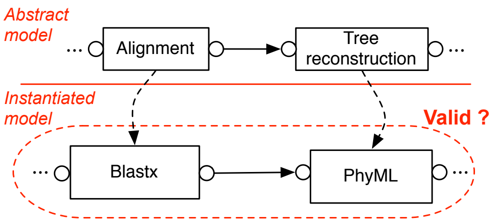

Figure 1: Problem.

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.

crete" 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 *my*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 subsection 4.2. From this categorization, we identify three compatibility situations, which we then put through a semi-automatic repair algorithm (cf. subsection 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:

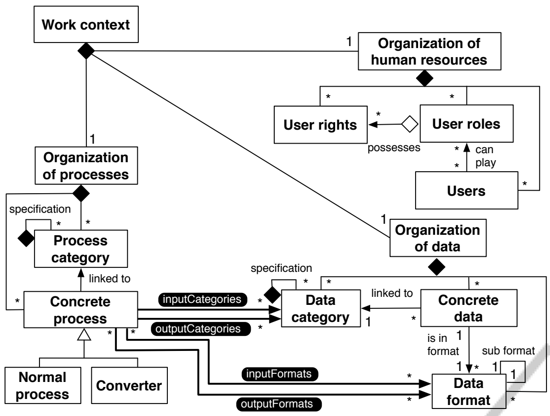

Figure 2: Work context.

- 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 categories, concrete data, and the various associated data formats[2].
- Organization of processes, for managing the description of process categories and concrete processes.

The concept of *Converter* introduced in the figure 2 refers to the concept of a specific process to convert between different formats of data belonging to the same data category[3].

To construct this environment, a simple formalism, designed for resource descriptions has been proposed and formalized by using XML schemas[4]. Figure 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 cardinality. 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.

To illustrate the concept of work context, we take a concrete example relating to the biological domain (cf. fig.4). The upper part of each hierarchy (processes and data) represents a set of categories (shown

[2]It should be noted that several data categories can share the same data format.

[3]This concept will be useful for resolving syntactic incompatibility.

[4]These XML Schema files are available online at http://www.lirmm.fr/ lin/project. However, we have not covered the organization of human resources.

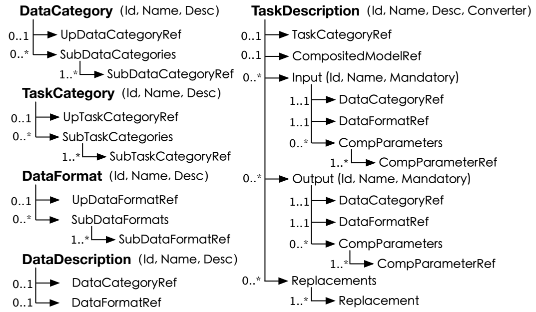

Figure 3: Schematic representation of the simple formalism designed for description of resources.

as ovals), ordered on the basis of the generalization/specialization relationship. The description of concrete resources (data or processes) are then associated with their category. A set of data formats (*bare, txt, Fasta, Tgf, Newick*) is also presented.

To take both the syntactic and semantic aspects into account, we propose the following formal description for the signature of every concrete process:

Name (list of Param_I) : (list of Param_O), with

- **Name** representing the name of the concrete process or operation.
- ***Param_I*** and ***Param_O*** which represent, respectively, 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 categories and formats. For example, the graphical description 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).*

4.2 Verification of Composition Compatibility in our Context

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

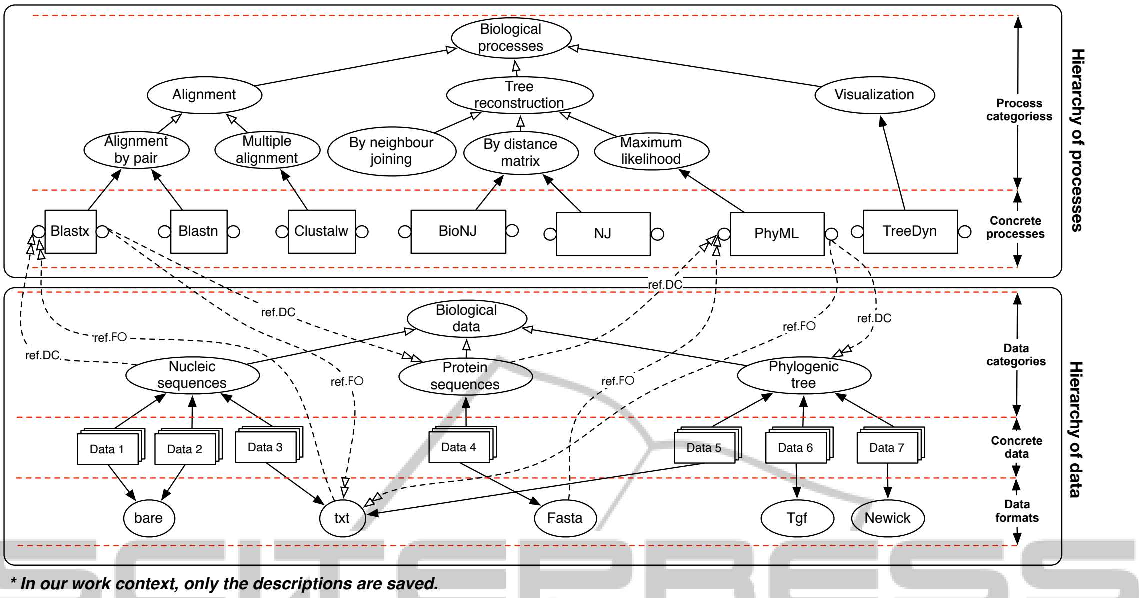

Figure 4: An example of a work context.

T1(dc1:fo1) : (dc2:fo2, dc3:fo3)
T2(dc4:fo4) : (dc5:fo5)

Supposing a link connecting *T1*'s output parameter *p1* (*dc3:fo3*) to *T2*'s input parameter *p2* (*dc4:fo4*). This composition $p1 \rightarrow p2$ allows us to define two types of compatibilities:

- **Syntactic Compatibility.** $p1 \rightarrow p2$ is syntactically compatible if *(fo3 = fo4)* $\vee$ *(fo3 is a sub-format of fo4)*, denoted $p1 \overset{Syn}{\rightarrow} p2$. Two parameters are syntactically compatible if they use the same data format or if they use an output format which is a sub-format of the input format. In the same way, $p1 \rightarrow p2$ is not compatible at the syntactical level if *(fo3* $\neq$ *fo4)* $\wedge$ *(fo3 is not a sub-format fo4)*, denoted $p1 \overset{Syn}{\nrightarrow} p2$.
- **Semantic Compatibility.** $p1 \rightarrow p2$ is semantically compatible if *(dc3 = dc4)* $\vee$ *(dc3 is a sub-category of dc4)*, denoted $p1 \overset{Sem}{\rightarrow} p2$. Two parameters are semantically compatible if they use the same category, or if they use an output category which is a sub-category of the input category. In the same way, $p1 \rightarrow p2$ is not compatible at the semantic level if *(dc3* $\neq$ *dc4)* $\wedge$ *(dc3 is not a sub-category of dc4)*, denoted $p1 \overset{Sem}{\nrightarrow} p2$.

From these two definitions, we identify three compatibility situations for the composition $p1 \rightarrow p2$:

- **Situation 1** $(p1 \overset{Sem}{\rightarrow} p2) \wedge (p1 \overset{Syn}{\rightarrow} p2)$**.** *p1* and *p2* are compatible at the semantic and syntactic levels. This is the ideal situation in our context, we designate it as valid.
- **Situation 2** $(p1 \overset{Sem}{\rightarrow} p2) \wedge (p1 \overset{Syn}{\nrightarrow} p2)$**.** *p1* and *p2* are compatible at the semantic level but not at the syntactic level. The composition is syntactically adaptable. An adaptation between the two data formats will be necessary (cf. converters).
- **Situation 3** $p1 \overset{Sem}{\nrightarrow} p2$**.** The two parameters are not semantically compatible. In such a case, it is pointless to proceed to verify their syntactic compatibility (in fact, for us, two parameters with different significations cannot be paired). The composition is semantically adaptable.

From these 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(p1, p2)*. 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
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.

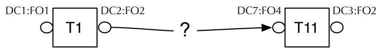

Figure 5: Initial composition.

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 $\mathbf{N} = \mathbf{N}_P \cup \mathbf{N}_{DC}$, with $\mathbf{N}_P$ being the set of process description nodes and $\mathbf{N}_{DC}$ being the set of data category nodes.
- A set of arcs **A**. If an arc $a=(n1,\ n2) \in \boldsymbol{A}$[5], then $(n1 \in \boldsymbol{N}) \wedge (n2 \in \boldsymbol{N}) \wedge (n1 \neq n2)$.
  Two types of arcs are present in the GRSEM, $\mathbf{A} = \mathbf{A}_R \cup \mathbf{A}_S$:
  1. $\mathbf{A}_R$ 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 $(n1 \in \mathbf{N}_P \wedge n2 \in \mathbf{N}_{DC}) \vee (n1 \in \mathbf{N}_{DC} \wedge n2 \in \mathbf{N}_P)$.
  2. $\mathbf{A}_S$ is a set of specialization arcs between data categories. If $a_s=(n1,\ n2) \in \boldsymbol{A}_S$, then $(n1 \in \mathbf{N}_{DC} \wedge n2 \in \mathbf{N}_{DC}) \wedge$ *(n1 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:

[5] From *n1* to *n2*.

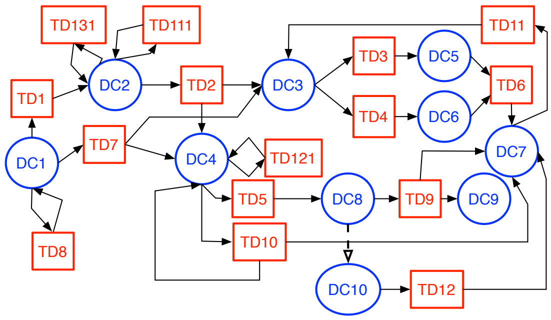

Figure 6: A GRSEM resource graph.

- $DC2 \mapsto TD2 \mapsto DC3 \mapsto TD3 \mapsto DC5 \mapsto TD6 \mapsto DC7$
- $DC2 \mapsto TD2 \mapsto DC3 \mapsto TD4 \mapsto DC6 \mapsto TD6 \mapsto DC7$
- $DC2 \mapsto TD2 \mapsto DC4 \mapsto TD10 \mapsto DC7$
- $DC2 \mapsto TD2 \mapsto DC4 \mapsto TD5 \mapsto DC8 \mapsto DC10 \mapsto TD12 \mapsto DC7$
- $DC2 \mapsto TD2 \mapsto DC4 \mapsto TD5 \mapsto DC8 \mapsto TD9 \mapsto DC7$

The user selects one of these paths, and all its intermediary processes are added to the instantiated workflow. Supposing the user chooses the itinerary $DC2 \mapsto TD2 \mapsto DC4 \mapsto TD5 \mapsto DC8 \mapsto DC10 \mapsto TD12 \mapsto DC7$, the intermediary processes *TD2*, *TD5* and *TD12* are added to the workflow (cf. figure 7). The input signature of *TD12* is modified to *(DC8:FO2)* because *DC8* is more specialized than *DC10* (as shown in figure 6).

Moreover, a set of links “tmpCategoryLink” indicating the semantic compatibilities of this composition is added to the workflow. The following stage consists of verifying the syntactic compatibility.

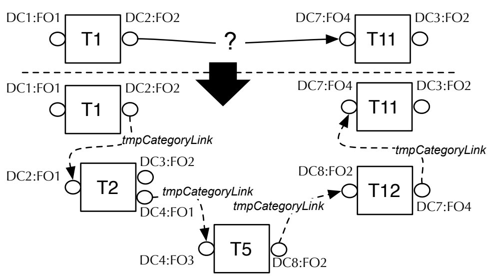

Figure 7: Modified composition 1.

4.3.2 Syntactic Adaptation

As already mentioned, syntactic adaptation consists of resolving syntactic incompatibility between two parameters of a composition. It is a matter of finding adaptations between different data formats present in the parameters. Note that this adaptation has a prerequisite: the composition concerned should already be semantically compatible.

To undertake this stage, a second, specific resource graph (GRSYN) is constructed using converters[6].

GRSYN is an oriented graph **GRSYN = (N, A)**, with:

- a set of nodes $\mathbf{N} = \mathbf{N}_{Comb} \cup \mathbf{N}_{Convert}$, with $\mathbf{N}_{Comb}$ the combined nodes, which designate the data category and the associated data format used by the converter (we will represent the node by $\frac{dc}{fo}$), and by $\mathbf{N}_{Convert}$ the converter nodes.
- a set of arcs **A**. An arc $a=(n1,\ n2) \in \boldsymbol{A}$ implies $(n1 \in \boldsymbol{N}_{Convert} \wedge n2 \in \boldsymbol{N}_{Comb}) \vee (n1 \in \boldsymbol{N}_{Comb} \wedge n2 \in \boldsymbol{N}_{Convert})$. 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.

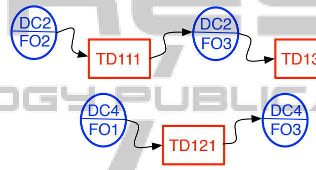

Figure 8: The GRSYN resource graph.

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 be found between *FO2* and *FO1*. A single itinerary was found in our GRSYN: $\frac{DC2}{FO2} \mapsto TD111 \mapsto \frac{DC2}{FO3} \mapsto TD131 \mapsto \frac{DC2}{FO1}$. If we retain this solution, the two converters *TD111* are *TD131* are substituted for the “tmpCategoryLink” link between *T1* and *T2*. In the same way, we can also establish syntactic adaptations for the composition $(DC4:FO1) \rightarrow (DC4:FO3)$ between *TD2* and *TD5*. The final updating of the instantiated workflow (cf. fig.9) corresponds to the replacing of the “tmpCategoryLink” links by the concerned converter(s).

[6]Note 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.

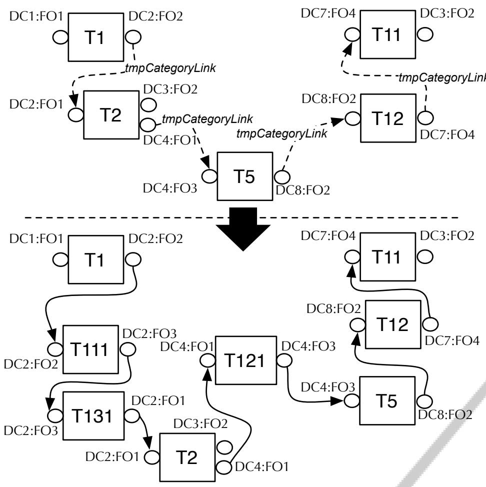

Figure 9: Modified composition 2.

According to this approach, the example in figure 1 will require only a syntactical adaptation of the composition *(ProteinSeq:txt)* $\rightarrow$ *(ProteinSeq:Fasta)* which can be achieved using a converter between the *txt* and *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 subcomponents:

(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[7].

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.

[7]For the time being, requests are constructed by taking into account the elements associated with each description of a resource.

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.

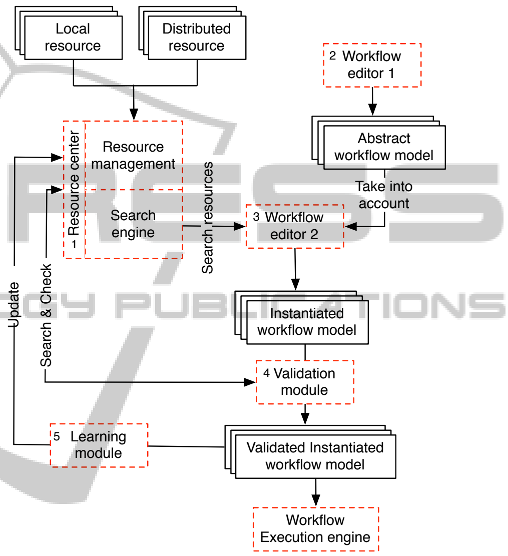

Figure 10: Functional presentation of the prototype.

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 futur works.

REFERENCES

Altintas, I., Ludäscher, B., Klasky, S., and Vouk, M. A. (2006). S04 - introduction to scientific workflow management and the kepler system. In *SC*, page 205.

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., and Weerawarana, S. (5 May 2003). Business process execution language for web services, version 1.1. http://www.ibm.com/developerworks/library/specification/ws-bpel/.

Beauche, S. and Poizat, P. (2008). Automated service composition with adaptive planning. In *ICSOC '08: Proceedings of the 6th International Conference on Service-Oriented Computing*, pages 530–537, Berlin, Heidelberg. Springer-Verlag.

Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001). *Web Services Description Language (WSDL) 1.1*. W3C, 1.1 edition.

CLARO, D. B. (2006). *SPOC - Un canevas pour la composition automatique de services web deédieés à la réalisation de devis*. PhD thesis, Université d'Angers.

Claro, D. B., Licchelli, O., Albers, P., and Macedo, J. D. A. (2008). Personalized reliable web service compositions. In *WONTO*.

Comerio, M., Truong, H.-L., Paoli, F., and Dustdar, S. (2009). Evaluating contract compatibility for service composition in the seco2 framework. In *Proceedings of the 7th International Joint Conference on Service-Oriented Computing*, ICSOC-ServiceWave '09, pages 221–236, Berlin, Heidelberg. Springer-Verlag.

Cunningham, S. and Denize, P. (1993). A tool for model generation and knowledge acquisition. In *Proc International Workshop on Artificial Intelligence and Statistics*, pages 213–222, Fort Lauderdale, Florida, USA.

Group, W. O. W. (2004). *OWL 2 : Web Ontology Language*. W3C.

Hamadi, R. and Benatallah, B. (2003). A petri net-based model for web service composition. In *Proceedings of the 14th Australasian database conference - Volume 17*, ADC '03, pages 191–200, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M., Li, P., and Oinn, T. (2006). Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34(Web-Server-Issue):729–732.

Joel Farrell, IBM Holger Lausen, D. I. (2007). *Semantic Annotations for WSDL and XML Schema*. W3C.

Julien Bourdon, P. B. and Fiorino, H. (2007). Architecture multi-agents pour la composition automatique de web services.

Kay, M. (23 January 2007). Xsl transformations (xslt) version 2.0. http://www.w3.org/TR/xslt.

Kiepuszewski, B., ter Hofstede, A., and van der Aalst, W. (2003). Fundamentals of control flow in workflows. *Acta Informatica*, 39:143–209.

Klusch, M. and Gerber, A. (2005). Semantic web service composition planning with owls-xplan. In *In Proceedings of the 1st Int. AAAI Fall Symposium on Agents and the Semantic Web*, pages 55–62.

Libourel, T., Lin, Y., Mougenot, I., Pierkot, C., and Desconnets, J.-C. (2010). A platform dedicated to share and mutualize environmental applications. In *ICEIS (1)*, pages 50–57.

Limthanmaphon, B. and Zhang, Y. (2003). Web service composition with case-based reasoning.

Lin, Y., Libourel, T., and Mougenot, I. (2009). A workflow language for the experimental sciences. In *ICEIS (3)*, pages 372–375.

Liu, Z., Ranganathan, A., and Riabov, A. (2007). Modelingweb services using semantic graph transformations to aid automatic composition. *Web Services, IEEE International Conference on*, 0:78–85.

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M. B., Lee, E. A., Tao, J., and Zhao, Y. (2006). Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. (2004). *OWL-S: Semantic Markup for Web Services*. W3C.

Milanovic, N. (2005). Contract-based web service composition framework with correctness guarantees. In Malek, M., Nett, E., and Suri, N., editors, *Service Availability*, volume 3694 of *Lecture Notes in Computer Science*, pages 52–67. Springer Berlin / Heidelberg.

Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F. (2003). Shop2: An htn planning system. *J. Artif. Intell. Res. (JAIR)*, 20:379–404.

Oinn, T., Greenwood, M., Addis, M., Alpdemir, N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M., Senger, M., Stevens, R., Wipat, A., and Wroe, C. (2006). Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice & Experience - Workflow in Grid Systems*.

Sirin, E., Parsia, B., Wu, D., Hendler, J. A., and Nau, D. S. (2004). Htn planning for web service composition using shop2. *J. Web Sem.*, 1(4):377–396.

van der Aalst, W. M. P. and ter Hofstede, A. H. M. (2005). Yawl: yet another workflow language. *Inf. Syst.*, 30(4):245–275.

W3C (2004). *RDF Primer*. W3C.