Environnement de workflow scientifique Validation et conformités

Lin Yuan, Isabelle Mougenot, Thérèse Libourel Rouge

Environnement de workflow scientifique

Validation et conformités

Yuan LIN, Isabelle MOUGENOT, Thérèse LIBOUREL

LIRMM, 161 rue Ada - Espace DEV, 500 rue JF Breton - Université de Montpellier II, Montpellier, France

RÉSUMÉ. *Les chercheurs des domaines environnementaux (biologie, information géographique, etc.) ont besoin de capitaliser, diffuser et valider les protocoles utilisés. Pour répondre à cette demande, le concept de workflow scientifique a été proposé. La plateforme MDWeb permet la collecte, la diffusion de ressources environnementales. Notre objectif est d'intégrer dans cette plateforme un environnement de workflow. L'article présente la notion d'environnement de workflow basé sur une architecture à trois niveaux (statique / intermédiaire / dynamique). Nous nous focalisons ensuite sur la phase de construction des protocoles expérimentaux (niveaux abstrait et concret) et sur la phase de validation des protocoles concrets en terme de vérification de la conformité de leurs compositions.*

ABSTRACT. *Scientists in environmental disciplines generally are required to capitalize, distribute and validate their experimental protocols in order to analyze complex multi-disciplinary problems. The concept of the scientific workflow provides a number of valuable solutions to address this requirement. Here we present a workflow environment that is integrated within MDWeb, a platform that allows users to collect and distribute environmental resources. In this manuscript we specifically introduce the concept of a workflow environment based on a three-tier architecture, consisting of static, intermediate and dynamic levels. Additionally we are focusing on both the construction phase of experimental protocols (in both abstract and concrete levels), and the validation phase of concrete protocols by addressing composition conformity.*

MOTS-CLÉS : *Environnement de workflow, composition des traitements, vérification de conformité*

KEYWORDS: *Workflow environment, process composition, conformity verification*

1. Introduction

1.1. *Généralités*

Les applications environnementales (biodiversité, écologie, agronomie, etc.) sont en pleine croissance. Les ressources (données et traitements) abondent et les initiatives pour les mettre à disposition des communautés scientifiques voient le jour sous l'impulsion de ces communautés ou de directives nationales ou internationales [INP 08, SER 08]. Les experts de ces domaines sont amenés à construire leurs propres expérimentations pour vérifier et valider leurs hypothèses. Pour construire ces expérimentations complexes, ils ont souvent besoin de rechercher les ressources adéquates et de les organiser ou réorganiser. De plus, chaque plan d'expérimentation mérite d'être sauvegardé afin de pouvoir être réexécuté plusieurs fois soit dans diverses configurations, soit sur diverses données de test. Dans ce contexte, nous pensons que l'utilisation de plateformes assistant le scientifique peut constituer une aide précieuse. Ces plateformes doivent répondre à divers enjeux : partage et mutualisation de ressources et de connaissances, aide à la construction, à la vérification et à la réutilisation de protocoles expérimentaux.

1.2. *Objectif*

Notre objectif est de construire un environnement d'expérimentation de workflow intégré dans une plateforme de partage et de mutualisation de données et traitements existantes [BAR 05]. Le workflow scientifique est une transposition du terme général de *workflow* [**?**] dans le contexte protocole expérimental, c'est-à-dire relatif uniquement à des enchaînements de traitements par l'intermédiaire de flots de données. Cette plateforme MDWeb[1] dont une vue générale est présentée figure 1, fournit une interface graphique aux utilisateurs, et plusieurs composants fonctionnels :

Un *gestionnaire de méta données* pour référencer données et traitements (via des descriptions de type méta données), un *moteur de recherche* pour localisation et consultation de ressources (données / traitements) (à partir des descriptions précédentes, et ceci pour des ressources locales ou distantes), un *environnement de workflow* pour définition, instanciation et exécution de chaînes de traitements expérimentales (composant qui est le sujet de notre recherche).

La section 2 présente les généralités autour de la vision d'un environnement de workflow à trois niveaux, ainsi que la problématique essentielle de vérification des protocoles expérimentaux avant exécution. La section 3 est dédiée à un panorama des outils et pistes de recherche existantes. La section 4 relate la proposition effectuée pour le contrôle de conformité. La section 5 définit les pistes de recherche futures.

---

1. Elle est open-source et a été développée suite à divers projets (Roselt / OSS, ACI Padoue, Projet rgional Syscolag).

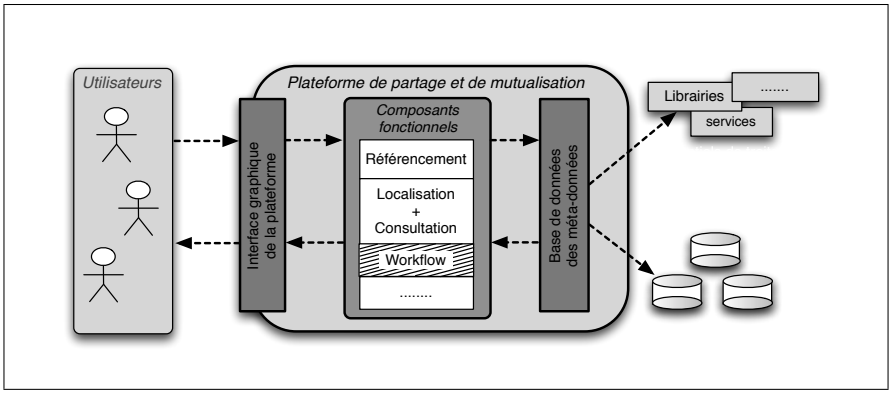

Figure 1 – Vue générale de la plateforme MDWeb

2. Environnement de workflow

2.1. *Généralités*

La réflexion menée a été guidée par le point de vue "métier" (expérimentateurs). Réaliser un protocole expérimental correspond au modèle général comportant trois étapes : 1) **Définition** : définition abstraite d'une chaîne de traitements correspondante à une expérimentation (planification d'expériences), 2) **Instanciation** : définition plus spécifique après identification des divers éléments de la chaîne (donnée / traitements), 3) **Exécution** : exécution personnalisée (selon des stratégies correspondantes aux besoins).

C'est à partir de ce cycle de vie expérimental et inspirés par les styles architecturaux proposés par OMG [OMG 06], que nous avons proposé pour le composant workflow la vision architecturale à 3-niveaux (cf. fig.2) :

1) Le niveau **statique** concerne la phase de conception. Il s'agit de construire des modèles de traitement métier (abstraits) à partir d'un langage simple.

2) Le niveau **intermédiaire** traduit une phase d'instanciation et de pré-contrôle. À partir du modèle du traitement métier, l'utilisateur construira la chaîne de traitements réelle en déterminant et localisant les traitements et les données les plus adéquates à l'expérimentation visée. Le pré-contrôle sera semi-automatisé (cf. section 4).

3) Le niveau **dynamique** concerne la phase d'exécution proprement dite. Celle-ci peut alors se dérouler à partir de différentes stratégies définies à la fois par l'expérimentateur et par les configurations opérationnelles.

Le niveau *statique* a été approfondi et étudié dans [LIN 09, LIB 10]. Il se base sur un langage défini par un méta modèle dont les éléments abstraits *tâches* ou *traitements* sont reliés par les liens unidirectionnels, et par l'intermédiaire de *port*. Pour faciliter la

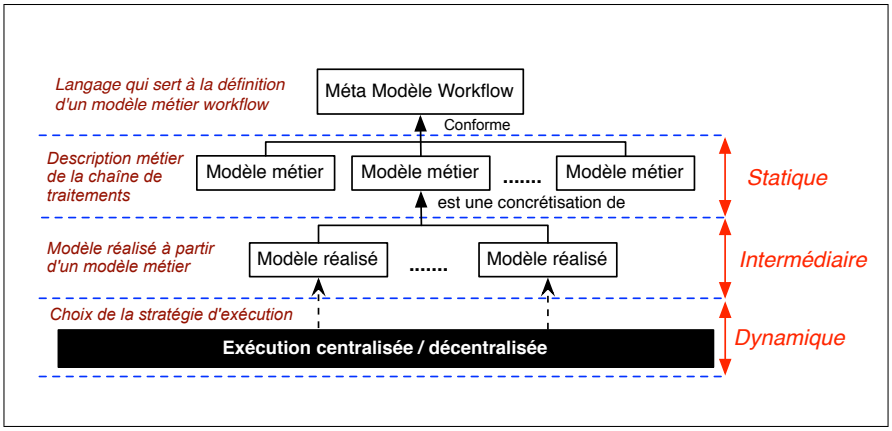

Figure 2 – Un environnement de workflow

manipulation des chaînes de traitements abstraits, un langage graphique correspondant a été proposé[2].

Nous nous intéresserons plus particulièrement dans cet article au niveau intermédiaire, et à la phase d'instanciation et de vérification de la validité du plan expérimental abstrait.

2.2. *Niveau Intermédiaire : problématique*

Au niveau intermédiaire, l'utilisateur transforme le modèle métier abstrait en modèle concret instancié à partir de données et traitements idoines (pour cela il utilisera le composant moteur de recherche de la plateforme et les méta-informations sur les ressources). Nous proposons de vérifier et *valider* le modèle concret avant de passer à l'exécution. Pour cela, nous allons exposer la problématique au travers d'un exemple illustratif issu d'un domaine biologique.

La figure 3 présente le protocole de traitements expérimental abstrait envisagé, ainsi que l'instanciation de celui-ci. Le protocole abstrait consiste à chaîner un alignement de séquences avec une reconstruction d'arbre afin de découvrir une des fonctions de la séquence étudiée. L'instanciation consiste à remplacer les éléments abstraits par des ressources réelles trouvées et localisées à l'aide du moteur de recherche de la plateforme. Cette recherche est basée sur des méta informations descriptives des ressources réelles (cf. section 4.1). Une difficulté supplémentaire se pose : les compositions entre traitements sont elles compatibles ? C'est ce problème que nous allons aborder, après un bref panorama des outils et approches existants.

2. Il sera utilisé dans les exemples ultérieurs

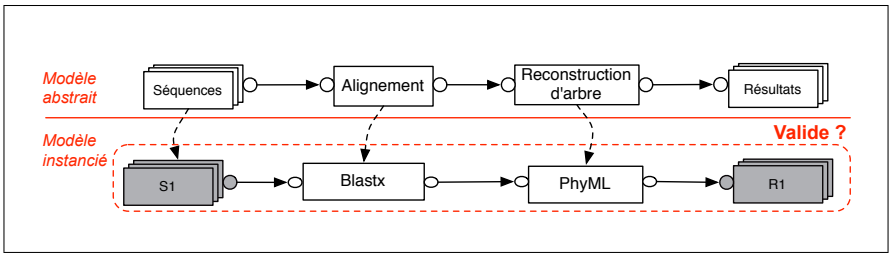

Figure 3 – Problématique

3. État de l'art sur la composition de traitements

L'analyse comportera deux parties : une autour des divers projets représentatifs du domaine, et l'autre précisant les différentes approches de recherche existantes dans ce domaine.

3.1. *Les environnements workflow*

Tous les environnements édudiés disposent d'une interface graphique, grâce à laquelle via des formalismes distincts, les scientifiques construisent leurs plans d'expérimentations. Cependant tous se situent au niveau que nous appelons "concret".

Kepler [LUD 06] est un environnement complet pour le workflow scientifique construit sur la plateforme Ptolemy II de l'Université de Berkeley. Dans cet environnement, les *acteurs* correspondent aux différents traitements et opérations envisageables, et ils sont dotés de *ports* qui représentent leurs paramètres d'entrée / sortie. Les compositions entre traitements sont faites interactivement par les scientifiques en reliant les *ports* d'*acteur* par des *chanels*. Le contrôle et l'orchestration de modèle de workflow est confié à des *directeurs*. Les adaptations nécessaires sont réalissées par des programmes intermédiaires (*senders* et *receivers*), qui assurent la compatibilité des données transférées sur un *chanel*.

Taverna [HUL 06] est un projet de workflow initialisé par l'équipe ^{my}Grid en Angleterre, utilisé principalement dans les domaines biologies. Les traitements dans cet environnement sont essentiellement des services web (auxquels peuvent s'adjoindre des librairies locales, des scripts manuscrits, etc.). Lors de la composition de traitements, l'utilisateur apparie manuellement les paramètres d'entrée / sortie des services web, ou invoque des *shim services*, adaptateurs spécifiques conçus antérieurement à partir d'appariements établis dans des expérimentations déjà construits et testés.

NetBeans est un environnement IDE généraliste, dont un des modules intégrés permet de construire un workflow par composition de services web en utilisant le langage BPEL (Business Process Execution Language) [AND 03]. Une bonne connaissance du standard BPEL est un pré-requis. La composition se fait par appariement ou

transformation manuelle entre éléments de messages échangés, ces règles d'appariement sont ensuite traduites à l'aide du langage XSLT (eXtensible Stylesheet Language Transformations) [KAY 07].

Weka [CUN 93] est une application du domaine apprentissage et fouille de données, réalisée par l'Université Waikato de Nouvelle Zélande. Elle comporte un composant **Weka KnowledgeFlow**, qui permet un chaînage de traitements relatifs aux expérimentations de fouille de données. Le modèle général de KnowledgeFlow obéit à la chaîne *Sélectionner les données* $\rightarrow$ *Filtrer* $\rightarrow$ *Classifier* $\rightarrow$ *Évaluer* $\rightarrow$ *Visualiser*. Grâce à l'interface graphique, les scientifiques concrétisent interactivement leurs expérimentations, et choisissent les convertisseurs pré-établis pour assurer la compatibilité de leurs workflows. L'environnement repose sur des catégories de données et d'algorithmes relatifs aux différents traitements précédemment construites.

3.2. *Les approches existantes*

Les approches que nous avons retenues sont essentiellement liées à la sémantique des traitements. Elles relèvent du domaine de l'intelligence artificielle.

L'approche ontologique Cet approche suppose l'existence d'ontologies de domaine préalablement construites pour les ressources (données / traitements), en utilisant des standards comme OWL[GRO 04]. Lors de la réalisation du workflow, l'utilisateur effectue la localisation des ressources et la composition des traitements guidée par ces ontologies. Par exemple, dans le projet METEOR-S proposé par l'université de Georgia (États Unis), le système workflow contrôle la compatibilité du chaînage des services web en utilisant l'extension SAWSDL [JOE 07] pour établir les relations entre les descriptions WSDL [CHR 01] de ces services web avec les concepts d'une ontologie OWL. Dans [LIU 07], les messages des services web sont exprimés sous forme de graphes RDF [W3C 04] la compatibilité est vérifiée par appariement entre ces graphes et les concepts de l'ontologie.

L'approche planificateur En intelligence artificielle, les planificateurs interviennent lorsque, pour atteindre un objectif fixé, on envisage de mettre en œuvre un plan d'actions. Dans un contexte workflow, les algorithmes planificateurs pourraient aider à trouver toutes les compositions de traitements possibles, pour obtenir, à partir d'une description d'état initial, l'état final souhaité. Les auteurs de l'article [BEA 08] utilisent deux structures spécifiques : *CSS (Capacity Semantic Structure)*, qui représente le plan du workflow sous forme d'arbre, les nœuds étant soit des traitements abstraits, soit des opérateurs de contrôles (séquence, choix ou parallélisme). *DSS (Data Semantic Structure)*, qui représente la structure des données admissibles pour chaque traitement. Le planificateur calcule toutes les possibilités de réalisation du workflow à partir du choix des services effectué par les utilisateurs. De nombreux plans peuvent être exhibés, qui prennent en compte l'adaptation des DSS concernées. Le plan retenu par l'utilisateur est transformé en YAWL orchestrators [AAL 05]. Un prototype relatif à cette approche a été implémenté (*GraphAdaptor*). L'article [KLU 05] utilise un en-

semble de descriptions de service web en OWL-S [MAR 04] et une ontologie OWL associée. Ceux-ci sont convertis en langage PDDL (Planning Domain Definition Language). Le planificateur *Xplan* peut à partir de ces traductions, calculer les planifications possibles pour atteindre un objectif défini initialement. L'article [SIR 04] montre de même l'usage du planificateur SHOP2 [NAU 03] dans la détermination de plan de compositions de services web (décrits en OWL-S). L'article [JUL 07] utilise une architecture multi-agent pour réaliser la planification de services web à partir d'une interaction entre agents (services) pour atteindre le but pré-fixé. L'article [CLA] utilise le système planificateur SPOC [CLA 06], qui détermine et ordonne les services web découverts dans la phase de localisation initiale. Il propose une optimisation de la planification guidée par le profil utilisateur.

Autres approches Les articles [LAJ 07, LIM 03] utilisent l'approche raisonnement par cas. La chaîne de traitements est établie après apprentissage sur des cas analogues (composition) et adaptation au contexte visé.

De cette analyse nous pouvons retirer la synthèse suivante : les travaux analysés se focalisent pour la plupart sur la composition des services web, les descriptions ontologiques s'avèrent primordiales pour détecter de l'incompatibilité sémantique, les adaptations nécessitent des transformations entre les structures de message incompatibles. L'approche planification n'est pas forcément naturelle, et peut s'avérer complexe et fastidieuse pour les experts non-informaticiens. Notre pré-occupation consiste à fournir un environnement dans lequel les chaînes de traitements peuvent invoquer des services web, mais aussi des librairies ou traitements spécifiques.

4. Notre approche

Compte tenu de la problématique évoquée en section 2.2, nous proposons donc une démarche basée sur l'analyse des compositions entre traitements d'un workflow guidée par la notion de contexte de travail (cf. sous-section 4.1). Nous définissons les différents types de compatibilité de composition dans la sous-section 4.2. À partir de cette catégorisation, nous identifions trois situations de compatibilité. Celles-ci nous amènerons aux propositions de réparations semi-automatiques (cf. sous-section 4.3).

4.1. *Contexte de travail*

La vérification de la validité d'un workflow consiste à vérifier la compatibilité de chaque composition de celui-ci en se basant sur la notion de contexte de travail. Ce contexte de travail (cf. fig.4) est composé de trois grandes organisations de descriptions de ressources, dénommées :

– Organisation des ressources humaines, qui gère la description des utilisateurs de la plateforme ainsi que celles de leurs différents rôles et droits d'accès associés,

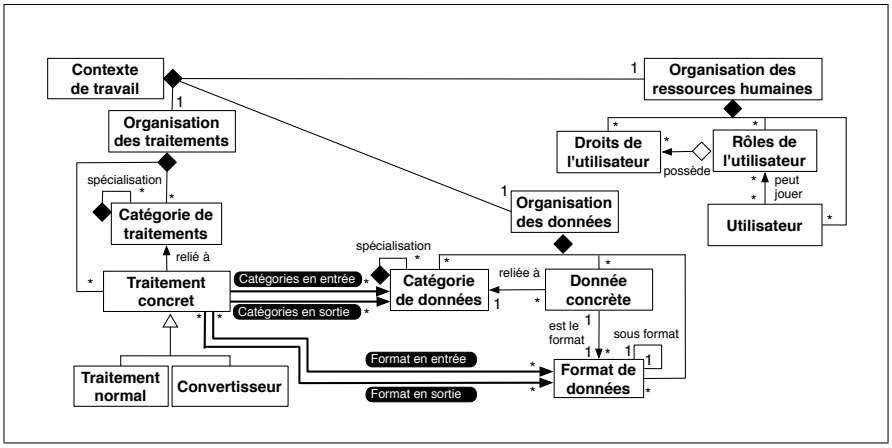

Figure 4 – Contexte de travail

– Organisation des données, qui gère la description des catégories de données, des données concrètes et des divers formats de données associés[3].

– Organisation des traitements, qui gère la description des catégories de traitements et des traitements concrets.

De notre point de vue, les catégories de données et de traitements relèvent d'une notion plus ouverte que celle de *Type*, car elle relève de la conceptualisation selon le domaine de l'utilisateur. Cela revient à dire que les hiérarchies de catégories de données et de traitements seront en relation étroite avec les ontologies des domaines expérimentaux concernés.

La notion de *Convertisseur* introduite dans la figure 4 correspond à la notion de traitement spécifique ayant en charge la réalisation d'adaptation entre différents formats relevant d'une même catégorie de données[4].

Pour la construction de cet environnement, un formalisme simple, dédié aux descriptions de ressources a été proposé et formalisé en utilisant XML schéma[5].

Pour illustrer la notion de contexte de travail, nous prenons un exemple concret relatif au domaine biologique (cf. fig.5). Le partie supérieure de chaque hiérarchie (traitements et données) représente un ensemble de catégories (représentées par des ovales), ordonné selon la relation de généralisation / spécialisation. Les descriptions des ressources concrètes (données ou traitements) sont ensuite associées à leur caté-

3. Remarque : Il est à noter que plusieurs catégories de données peuvent partager le même format.
4. Cette notion sera utile lors de la résolution de l'incompatibilité syntaxique.
5. Ces fichiers XML Schémas sont mis en ligne, et accessibles sur la page `http://www.lirmm.fr/~lin/project`. Nous n'avons cependant pas détaillé l'organisation de ressources humaines.

gorie. Un ensemble de formats de données (*bare, txt, Fasta, Tgf, Newick*) est aussi présenté.

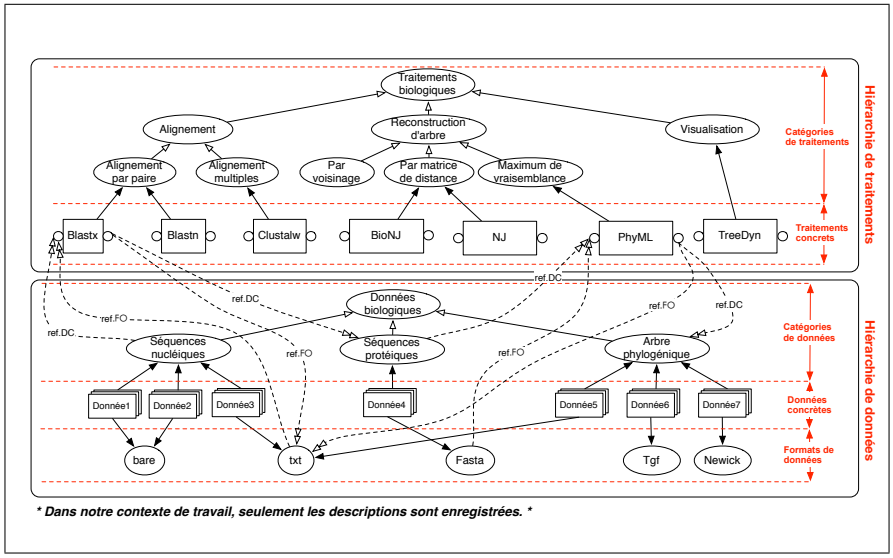

Figure 5 – Un exemple de contexte de travail

Pour prendre en compte les deux aspects syntaxique et sémantique, nous proposons la description formelle suivante pour la signature de tout traitement concret :

Nom (list de Param_E) : (list de Param_S), avec

– **Nom** qui représente le nom du traitement ou de l'opération concrète.

– ***Param_E*** et ***Param_S*** qui représente un des paramètres d'entrée ou de sortie ***p***. ***p*** est de la forme ***(dc :fo)***, avec ***dc*** et ***fo*** relatifs aux catégorie et format de données utilisés.

Graphiquement, les deux aspects syntaxique et sémantique de chaque signature sont matérialisés par les flèches pointillées, qui relient les traitements concrets, leurs catégories et formats de données. Par exemple, la description graphique du traitement *Blastx* précise que ce traitement possède un paramètre en entrée et un en sortie. Les flèches étiquetées *ref.DC* et *ref.FO* relient ces paramètres (entrée et sortie) aux catégorie et format de données associés. Finalement, la signature du traitement *Blastx* peut être représentée comme

Blastx*(SéqNucléiques :txt) :(SéqProtéiques :txt).*

4.2. *Vérification de la compatibilité de composition dans notre contexte*

Vérifier la conformité de composition d'un workflow avant exécution consiste à détecter et corriger les incompatibilités de chaque composition de celui-ci. Plus précisément, il s'agit de vérifier la compatibilité des deux extrémités (paramètres) de chaque lien. La description formelle des signatures proposée pour les traitements nous permet de définir les notions de compatibilités syntaxique et sémantique. Soit deux traitements *T1* et *T2* décrits par les signatures suivantes :

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

Supposons un lien reliant le paramètre *p1* (*dc3 :fo3*) de sortie de *T1* au paramètre *p2* (*dc4 :fo4*) d'entrée de *T2*. Cette composition $p1 \rightarrow p2$ nous permet la définition de deux types de compatibilités :

– **Compatibilité syntaxique :** $p1 \rightarrow p2$ est compatible syntaxiquement si *(fo3 = fo4)* $\vee$ *(fo3 est un sous-format de fo4)*, noté $p1 \overset{Syn}{\rightarrow} p2$. Deux paramètres sont compatibles syntaxiquement s'ils utilisent le même format de données, ou s'ils utilisent en sortie un sous-format de celui utilisé en entrée. De la même manière, $p1 \rightarrow p2$ n'est pas compatible au niveau syntaxique si *(fo3* $\neq$ *fo4)* $\wedge$ *(fo3 n'est pas un sous-format de fo4)*, noté $p1 \overset{Syn}{\nrightarrow} p2$.

– **Compatibilité sémantique :** $p1 \rightarrow p2$ est compatible sémantiquement si *(dc3 = dc4)* $\vee$ *(dc3 est une sous-catégorie de dc4)*, noté $p1 \overset{Sem}{\rightarrow} p2$. Deux paramètres sont compatibles sémantiquement s'ils utilisent la même catégorie, ou s'ils utilisent en sortie une sous-catégorie de celle utilisée en entrée. De la même manière, $p1 \rightarrow p2$ n'est pas compatible au niveau sémantique si *(dc3* $\neq$ *dc4)* $\wedge$ *(dc3 n'est pas une sous-catégorie de dc4)*, noté $p1 \overset{Sem}{\nrightarrow} p2$.

À partir de ces deux définitions, nous identifions trois situations de compatibilité pour la composition $p1 \rightarrow p2$:

– **Situation 1** $(p1 \overset{Sem}{\rightarrow} p2) \wedge (p1 \overset{Syn}{\rightarrow} p2)$: *p1* et *p2* sont compatibles aux niveaux sémantique et syntaxique. C'est la situation idéale dans notre contexte, nous la désignons comme valide.

– **Situation 2** $(p1 \overset{Sem}{\rightarrow} p2) \wedge (p1 \overset{Syn}{\nrightarrow} p2)$: *p1* et *p2* sont compatibles au niveau sémantique, mais pas au niveau syntaxique. La composition est adaptable syntaxiquement. Une adaptation entre deux formats de données sera nécessaire (cf. les convertisseurs).

– **Situation 3** $p1 \overset{Sem}{\nrightarrow} p2$: Les deux paramètres ne sont pas compatibles sémantiquement. Dans ce cas-là, il est donc inutile de continuer à vérifier leur compatibilité syntaxique (en effet pour nous, deux paramètres de significations différentes ne peuvent être appariés). La composition est adaptable sémantiquement.

A partir de ces définitions, nous développons l'approche proposée pour résoudre les incompatibilités.

4.3. *Réparation d'une composition incompatible*

Parmi les trois situations déclinées précédemment, les deux dernières nécessitent des adaptations supplémentaires avant de passer à la phase d'exécution.

Une procédure générale permet la vérification de la validité d'une composition du workflow. Elle déclenche deux types d'adaptations : l'adaptation sémantique pour l'incompatibilité sémantique et l'adaptation syntaxique pour l'incompatibilité syntaxique.

Pour illustrer notre approche, un jeu de données exemple a été créé. Il comporte 10 définitions de catégorie de données et 4 formats de données intégrés, ainsi que 14 descriptions de traitements, dont trois convertisseurs *TD111*, *TD121* et *TD131*. Leur signature sont :

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)

Prenons la composition entre *T1* et *T11*, (DC2 :FO2) $\rightarrow$ (DC7 :FO4), elle correspond à la *Situation 3*. Pour valider cette composition, nous devons donc trouver une solution pour assurer premièrement la compatibilité sémantique, puis dans un deuxième temps, pour assurer la compatibilité syntaxique.

Ces deux adaptations successives nécessitent la définition et la construction de deux types de graphes de ressources construits à partir du contexte de travail (GRSEM et GRSYN).

4.3.1. *Adaptation Sémantique*

Pour une composition sémantiquement incompatible, la solution proposée consiste à trouver les traitements ou les compositions de traitements qui permettent de transformer la catégorie de données source vers celle de destination. Pour réaliser ce premier objectif, nous construisons le graphe de ressources (*GRSEM*).

GRSEM est un graphe orienté **GRSEM = (N, A)**, avec :

– Un ensemble de nœuds $\mathbf{N} = \mathbf{N}_T \cup \mathbf{N}_{DC}$, avec $\mathbf{N}_T$ ensemble des nœuds descriptions de traitements, et $\mathbf{N}_{DC}$ ensemble des nœuds catégories de données.

– Un ensemble d'arcs **A**. Si un arc ***a=(n1, n2)*** $\in \boldsymbol{A}$[6], alors $(n1 \in \boldsymbol{N}) \wedge (n2 \in \boldsymbol{N}) \wedge (n1 \neq n2)$.

Deux types d'arcs sont présents dans GRSEM, $\mathbf{A} = \mathbf{A}_R \cup \mathbf{A}_S$:

1) $\mathbf{A}_R$ est l'ensemble des arcs de référence vers les catégories de données utilisées par un paramètre de traitement. Si a_r*=(n1, n2)* $\in A_R$, alors *(n1*$\in \boldsymbol{N}_T \wedge$ *n2*$\in \boldsymbol{N}_{DC}$*)* $\vee$ *(n1*$\in \boldsymbol{N}_{DC} \wedge$ *n2*$\in \boldsymbol{N}_T$*)*.

2) $\mathbf{A}_S$ est l'ensemble des arcs de spécialisation entre les catégories de données. Si a_s*=(n1, n2)* $\in \boldsymbol{A}_S$, alors *(n1*$\in \boldsymbol{N}_{DC} \wedge$ *n2*$\in \boldsymbol{N}_{DC}$*)* $\wedge$ *(n1 représente une sous-catégorie directe de celle représentée par n2)*.

6. De *n1* vers *n2*

Le GRSEM de la figure 6 a été généré à partir du jeu de données exemple : les nœuds circulaires représentent les catégories de données, les nœuds rectangulaires correspondent aux descriptions de traitements. Les arcs de références et de spécialisations sont ensuite ajoutés entre les nœuds.

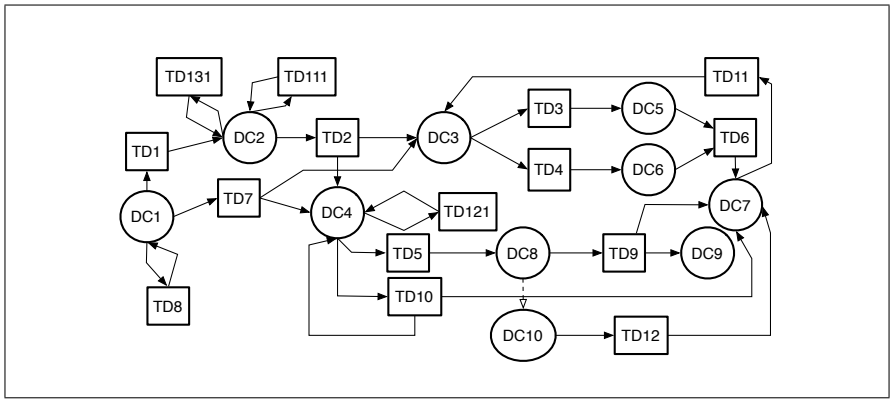

Figure 6 – Un graphe de ressources GRSEM

L'adaptation sémantique peut être considérée comme un problème de recherche de chemins entre deux nœuds (de catégorie de données) dans le graphe de ressources GRSEM.

Un algorithme récursif est utilisé, il prend en entrée les nœuds de deux catégories de données concernées, et génère tous les chemins possibles entre eux en parcourant GRSEM. Chaque chemin trouvé contient un ensemble de nœuds intermédiaires, et représente en fait une adaptation sémantique potentielle (séquence de traitements intermédiaires). Cette adaptation se situant au niveau *catégorie*, c'est-à-dire au niveau *concept*, elle ne suffira pas pour déterminer la validation de la chaîne.

Pour la composition (DC2 :FO2) $\rightarrow$ (DC7 :FO4), et le graphe GRSEM construit, nous obtenons les adaptations potentielles suivantes :

– $DC2 \mapsto TD2 \mapsto DC3 \mapsto TD3 \mapsto DC5 \mapsto TD6 \mapsto DC7$

– $DC2 \mapsto TD2 \mapsto DC3 \mapsto TD4 \mapsto DC6 \mapsto TD6 \mapsto DC7$

– $DC2 \mapsto TD2 \mapsto DC4 \mapsto TD10 \mapsto DC7$

– $DC2 \mapsto TD2 \mapsto DC4 \mapsto TD5 \mapsto DC8 \mapsto DC10 \mapsto TD12 \mapsto DC7$

– $DC2 \mapsto TD2 \mapsto DC4 \mapsto TD5 \mapsto DC8 \mapsto TD9 \mapsto DC7$

L'utilisateur choisit un de ces chemins, et tous les traitements intermédiaires inclus dans celui-ci sont ajoutés au workflow instancié. Supposons que l'utilisateur choisisse l'itinéraire $DC2 \mapsto TD2 \mapsto DC4 \mapsto TD5 \mapsto DC8 \mapsto DC10 \mapsto TD12 \mapsto DC7$, les traitements intermédiaires *TD2*, *TD5* et *TD12* sont ajoutés au workflow (cf. figure 7). La signature d'entrée de *TD12* est modifiée en *(DC8 :FO2)* car *DC8* est plus spécialisée que *DC10* (comme précisé dans la figure 6). De plus un ensemble de liens "tmpCategoryLink" indiquant les compatibilités sémantiques de cette composition est ajouté au workflow. L'étape suivante consiste à vérifier la compatibilité syntaxique.

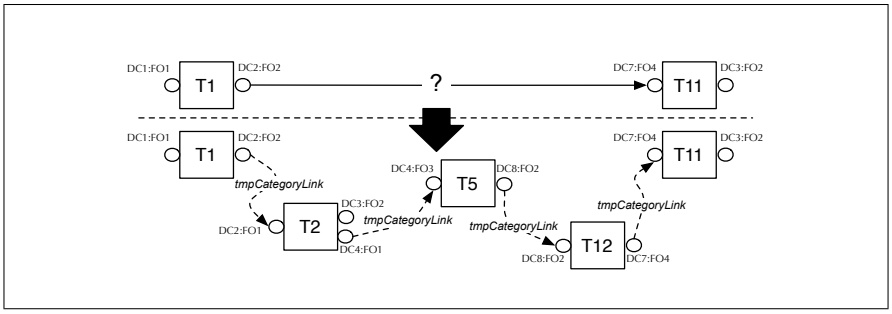

Figure 7 – Composition modifiée 1

4.3.2. *Adaptation syntaxique*

Comme introduit précédemment, l'adaptation syntaxique consiste à résoudre l'incompatibilité syntaxique entre deux paramètres d'une composition. Il s'agit de trouver les adaptions entre différents formats de données présents dans les paramètres. Remarque : cette adaptation a pour pré-requis le fait que la composition concernée soit sémantiquement compatible.

Pour réaliser cette étape, un deuxième graphe de ressources spécifique (GRSYN) est construit à partir des convertisseurs[7].

GRSYN est un graphe orienté **GRSYN = (N, A)**, avec :

– un ensemble de nœuds $\mathbf{N} = \mathbf{N}_{Comb} \cup \mathbf{N}_{Convert}$, avec $\mathbf{N}_{Comb}$ les nœuds combinés, qui désignent la catégorie de données et le format de données associés utilisés par le convertisseur (on représentera le nœud par $\frac{dc}{fo}$), et $\mathbf{N}_{Convert}$ les nœuds convertisseurs.

– un ensemble d'arcs **A**. Un arc ***a=(n1, n2)*** $\in \boldsymbol{A}$ implique $(n1 \in \boldsymbol{N}_{Convert} \wedge n2 \in \boldsymbol{N}_{Comb}) \vee (n1 \in \boldsymbol{N}_{Comb} \wedge n2 \in \boldsymbol{N}_{Convert})$.. Cet ensemble correspond aux liens de références entre un nœud convertisseur et un nœud combiné.

À partir du jeu de données exemple, le GRSYN généré est présenté la figure 8.

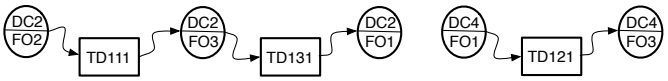

Figure 8 – Le graphe de ressources GRSYN

Comme précédemment, l'adaptation syntaxique peut être considérée comme une recherche de chemins dans le GRSYN. Reprenons la composition entre *T1* et *T11*, après une première

7. Remarque, un convertisseur pour nous, comme défini précédemment est un traitement spécifique qui transforme les données d'une même catégorie de données mais ayant des formats différents. On suppose donc que ces convertisseurs existent.

phase d'adaptation sémantique, nous avons obtenu un nouveau modèle (cf. fig.7) qui est compatible sémantiquement sur toutes ces compositions. Il reste à vérifier la compatibilité syntaxique de chaque lien "tmpCategoryLink". Si on prend le lien entre *T1* et *T2*, les deux paramètres reliés sont (DC2 :FO2) et (DC2 :FO1), il faut donc chercher une adaptation syntaxique entre *FO2* et *FO1*. Un seul itinéraire a été trouvé dans notre GRSYN : $\frac{DC2}{FO2} \mapsto$ *TD111* $\mapsto \frac{DC2}{FO3} \mapsto$ *TD131* $\mapsto \frac{DC2}{FO1}$. En retenant cette proposition, les deux convertisseurs *TD111* et *TD131* sont substitués au lien "tmpCategoryLink" entre *T1* et *T2*. De la même manière, nous pouvons aussi établir les adaptations syntaxiques pour la composition *(DC4 :FO1)* $\rightarrow$ *(DC4 :FO3)* entre *TD2* et *TD5*. La mise à jour complète du workflow instancié (cf. fig.9) correspond aux remplacements des liens "tmpCategoryLink" par le ou les convertisseurs concernés.

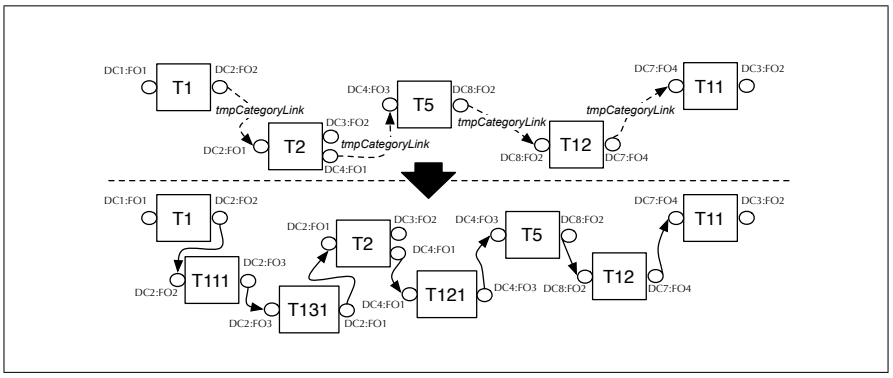

Figure 9 – Composition modifiée 2

Selon cette approche, l'exemple de la figure 3 nécessiterait uniquement une adaptation syntaxique de la composition *(SéqProtéique :txt)* $\rightarrow$ *(SéqProtéique :Fasta)* réalisable par un convertisseur entre les deux formats *txt* et *Fasta*.

L'approche formelle a été testée via un prototype (implémenté en Java). Une démonstration du prototype est en ligne (`http://www.lirmm.fr/~lin/project`).

5. Conclusion et perspectives

L'insertion d'un composant de workflow dans une plateforme de partage et de mutualisation nous parait importante, cependant le problème d'incompatibilité constitue une des problématiques majeures de la composition de traitements. La solution proposée nous semble judicieuse, les recherches ultérieures envisagées portent sur l'extension des descriptions de ressources et sur l'insertion du composant du workflow au sein de la plateforme, afin de le coupler avec un moteur d'exécution. Les diverses approches concernant la vérification du contrat de composition [COM 09, MIL 05], le contrôle sur les comportements des composants basé sur le réseau de pétri [KIE 03, HAM 03] traitent les différents aspects de la composition et proposent les pistes intéressantes, elles nous serviront effectivement dans la recherche future.

En ce qui concerne l'extension des descriptions, nous pensons nous orienter vers des formalismes plus riches tels WSDL ou OWL-S. L'aspect sémantique des descriptions de ressources

pourrait ainsi être complété. La construction du contexte de travail pourrait bénéficier de l'apport d'ontologies issues des domaines expérimentaux visés. La sémantique des traitements est pour l'instant portée uniquement par le nom et les catégories de données des paramètres, on peut envisager d'étendre celle-ci en utilisant des relations terminologiques (synonymie, etc.), ainsi qu'en ajoutant aux descriptions des compléments relatifs aux comportement du traitement (machine d'état par exemple).

6. Bibliographie

[AAL 05] VAN DER AALST W. M. P., TER HOFSTEDE A. H. M., « YAWL : yet another workflow language », *Inf. Syst.*, vol. 30, n° 4, 2005, p. 245-275.

[AND 03] ANDREWS T., CURBERA F., DHOLAKIA H., GOLAND Y., KLEIN J., LEYMANN F., LIU K., ROLLER D., SMITH D., THATTE S., TRICKOVIC I., WEERAWARANA S., « Business Process Execution Language for Web Services, Version 1.1 », , 5 May 2003.

[BAR 05] BARDE J., LIBOUREL T., MAUREL P., « A Metadata Service for Integrated Management of Knowledges Related to Coastal Areas », *Multimedia Tools Appl.*, vol. 25, n° 3, 2005, p. 419-429.

[BEA 08] BEAUCHE S., POIZAT P., « Automated Service Composition with Adaptive Planning », *ICSOC '08 : Proceedings of the 6th International Conference on Service-Oriented Computing*, Berlin, Heidelberg, 2008, Springer-Verlag, p. 530–537.

[CHR 01] CHRISTENSEN E., CURBERA F., MEREDITH G., WEERAWARANA S., « Web Services Description Language (WSDL) 1.1 », W3C, 1.1 édition, 15 March 2001.

[CLA] CLARO D. B., LICCHELLI O., ALBERS P., MACEDO J. D. A., « Personalized Reliable Web service Compositions ».

[CLA 06] CLARO D. B., « SPOC - Un canevas pour la composition automatique de services web deédieés à la réalisation de devis », PhD thesis, Université d'Angers, 2006.

[COM 09] COMERIO M., TRUONG H.-L., PAOLI F., DUSTDAR S., « Evaluating Contract Compatibility for Service Composition in the SeCO2 Framework », *Proceedings of the 7th International Joint Conference on Service-Oriented Computing*, ICSOC-ServiceWave '09, Berlin, Heidelberg, 2009, Springer-Verlag, p. 221–236.

[CUN 93] CUNNINGHAM S., DENIZE P., « A Tool for Model Generation and Knowledge Acquisition », *Proc International Workshop on Artificial Intelligence and Statistics*, Fort Lauderdale, Florida, USA, 1993, p. 213-222.

[GRO 04] GROUP W. O. W., « OWL 2 : Web Ontology Language », W3C, February 2004.

[HAM 03] HAMADI R., BENATALLAH B., « A Petri net-based model for web service composition », *Proceedings of the 14th Australasian database conference - Volume 17*, ADC '03, Darlinghurst, Australia, Australia, 2003, Australian Computer Society, Inc., p. 191–200.

[HUL 06] HULL D., WOLSTENCROFT K., STEVENS R., GOBLE C., POCOCK M., LI P., OINN T., « Taverna : a tool for building and running workflows of services », , 2006.

[INP 08] INPIRE, « INSPIRE metadata implementing rules based on ISO 19115 and ISO 19119 », European Comission, 2008.

[JOE 07] JOEL FARRELL IBM HOLGER LAUSEN D. I., « Semantic Annotations for WSDL and XML Schema », W3C, August 2007.

[JUL 07] JULIEN BOURDON P. B., FIORINO H., « Architecture multi-agents pour la composition automatique de web services », 2007.

[KAY 07] KAY M., « XSL Transformations (XSLT) Version 2.0 », , 23 January 2007.

[KIE 03] KIEPUSZEWSKI B., TER HOFSTEDE A., VAN DER AALST W., « Fundamentals of control flow in workflows », *Acta Informatica*, vol. 39, 2003, p. 143-209, Springer Berlin / Heidelberg.

[KLU 05] KLUSCH M., GERBER A., « Semantic web service composition planning with OWLS-XPlan », *In Proceedings of the 1st Int. AAAI Fall Symposium on Agents and the Semantic Web*, 2005, p. 55–62.

[LAJ 07] LAJMI S., GHEDIRA C., GHEDIRA K., « Une méthode d'apprentissage pour la composition de services web », *L'objet. Volume 8 - n2/2007*, vol. 8, 2007.

[LIB 10] LIBOUREL T., LIN Y., MOUGENOT I., PIERKOT C., DESCONNETS J.-C., « A Platform Dedicated to Share and Mutualize Environmental Applications », *ICEIS (1)*, 2010, p. 50-57.

[LIM 03] LIMTHANMAPHON B., ZHANG Y., « Web Service Composition with Case-Based Reasoning », 2003.

[LIN 09] LIN Y., LIBOUREL T., MOUGENOT I., « A Workflow Language for the Experimental Sciences », *ICEIS (3)*, 2009, p. 372-375.

[LIU 07] LIU Z., RANGANATHAN A., RIABOV A., « ModelingWeb Services using Semantic Graph Transformations to aid Automatic Composition », *Web Services, IEEE International Conference on*, vol. 0, 2007, p. 78-85, IEEE Computer Society.

[LUD 06] LUDÄSCHER B., ALTINTAS I., BERKLEY C., HIGGINS D., JAEGER E., JONES M. B., LEE E. A., TAO J., ZHAO Y., « Scientific workflow management and the Kepler system », *Concurrency and Computation : Practice and Experience*, vol. 18, n° 10, 2006, p. 1039-1065.

[MAR 04] MARTIN D., BURSTEIN M., HOBBS J., LASSILA O., MCDERMOTT D., MCILRAITH S., NARAYANAN S., PAOLUCCI M., PARSIA B., PAYNE T., SIRIN E., SRINIVASAN N., SYCARA K., « OWL-S : Semantic Markup for Web Services », W3C, 22 November 2004.

[MIL 05] MILANOVIC N., « Contract-Based Web Service Composition Framework with Correctness Guarantees », MALEK M., NETT E., SURI N., Eds., *Service Availability*, vol. 3694 de *Lecture Notes in Computer Science*, p. 52-67, Springer Berlin / Heidelberg, 2005.

[NAU 03] NAU D. S., AU T.-C., ILGHAMI O., KUTER U., MURDOCK J. W., WU D., YAMAN F., « SHOP2 : An HTN Planning System », *J. Artif. Intell. Res. (JAIR)*, vol. 20, 2003, p. 379-404.

[OMG 06] OMG, *Meta Object Facility (MOF) Core Specification, Version 2.0, formal/06-01-01*, January 2006.

[SER 08] SERVICES" D. T. N., « Draft Implementing Rules for Discovery Services (IR3) », 2008.

[SIR 04] SIRIN E., PARSIA B., WU D., HENDLER J. A., NAU D. S., « HTN planning for Web Service composition using SHOP2 », *J. Web Sem.*, vol. 1, n° 4, 2004, p. 377-396.

[W3C 04] W3C, « RDF Primer », W3C, February 2004.