A Formal Approach to Support Interoperability in Scientific Meta-workflows

Arshad, J., Terstyanszky, G., Kiss, T., Weingarten, N. and Taffoni, G.

This is an author's accepted manuscript of an article published in Journal of Grid Computing, 14 (4), pp. 655-671, 2016.

The final publication is available at Springer via:

https://dx.doi.org/10.1007/s10723-016-9383-x

A Formal Approach to Support Interoperability in Scientific Meta-workflows

Junaid Arshad[α], Gabor Terstyanszky[α], Tamas Kiss[α], Noam Weingarten[α], Giuliano Taffoni[γ]
[α]Center for Parallel Computing, University of Westminster, London, UK
(G.Z.Terstyanszky, T.Kiss, Weingan)@westminster.ac.uk
[γ]Astronomical Observatory of Trieste, Trieste Italy

Corresponding Author: Junaid Arshad
Email: rjarshad@gmail.com

***Abstract* — Scientific workflows orchestrate the execution of complex experiments frequently using distributed computing platforms. Meta-workflows represent an emerging type of such workflows which aim to reuse existing workflows from potentially different workflow systems to achieve more complex and experimentation minimizing workflow design and testing efforts. Workflow interoperability plays a profound role in achieving this objective. This paper is focused at fostering interoperability across meta-workflows that combine workflows of different workflow systems from diverse scientific domains. This is achieved by formalizing definitions of meta-workflow and its different types to standardize their data structures used to describe workflows to be published and shared via public repositories. The paper also includes thorough formalization of two workflow interoperability approaches based on this formal description: the coarse-grained and fine-grained workflow interoperability approach. The paper presents a case study from Astrophysics which successfully demonstrates the use of the concepts of meta-workflows and workflow interoperability within a scientific simulation platform.**

Keywords—Meta-workflows; Workflow Interoperability; Science Gateways; Workflow Repository.

This research has been funded by the EU-FP7 funding programme for research, technological development and demonstration under ER-FLOW project with grant agreement no. 312579

I. Introduction

Scientific workflows represent complex computational experiments conducted by scientists focused at identifying and addressing scientific problems across diverse subject domains. Such experiments commonly involve analyzing large volumes of data and can be run on a variety of computing platforms including high performance computing infrastructures such as clusters, grids and clouds. Scientific workflows are often composed of control and data flow statements and rules which perform the analytics required to achieve the intended experimentation [1]. A typical scientific workflow is composed of one or more individual tasks each of which requires certain number of inputs and generates the respective output(s) after performing its intended function.

An interesting and emerging trend in workflow development is composing workflows by integrating one or more existing workflows called *sub-workflows* or *embedded workflows* into *meta-workflows*. These complex workflows, may utilize existing workflows incorporating them as components of the meta-workflow for faster and more efficient development. Meta-workflows engage complex orchestration of applications which may span across multiple domains and include workflows from heterogeneous workflow platforms. For such complex workflows the nodes represent a combination of workflow jobs and also sub-workflows which can host multiple tasks within them.

The use of meta-workflows to achieve complex scientific experimentation has been growing consistently in various scientific domains. For instance, within the MoSGrid Science Gateway [2], meta-workflows have been used extensively especially in docking and molecular dynamics domains contributing to facilitated job submission and output analysis. Furthermore, recent advancements in Heliophysics [3] have witnessed remarkable growth in use of meta-workflows to automate manually cumbersome and complex tasks to aid studies focused at analyzing the impact of solar wind on the Earth's atmosphere. Workflow systems have been widely used also in Astronomy and Astrophysics, and recently meta-workflows have been used extensively to facilitate access to Astronomical catalogues and archives. In particular, Astronomers implemented a library of simple atomic operations that can be re-used as basic components (sub-workflows) of other more complex meta-workflows that represent Astronomical use cases [4]. This paper describes a scientific use case from Astrophysics to highlight the significance of the contribution made by the paper.

Our focus in this paper is to investigate the challenges encountered in facilitating widespread sharing of such complex scientific meta-workflows. The work presented in this paper is an extension to our work in [5] and is aimed at the formal definition and analysis of the different types of meta-workflows. Moreover, the paper highlights the role of workflow interoperability in different approaches for meta-workflow creation and execution. In particular, we present a detailed formal description of meta-workflows to support their design and execution. To the best of our knowledge, this work demonstrates a pioneering effort for formal definition of different types of meta-workflows and approaches for their creation and execution. Therefore, we are making significant contribution towards scientific workflow interoperability by providing a unified method to define such workflows. Within this context, we highlight current limitations of workflow repositories when describing and managing meta-workflows and propose the adoption of our formal approach for describing meta-workflows to overcome these limitations. We demonstrate the effectiveness of our proposed approach by including a scientific use-case from the Astrophysics domain which exemplifies the formal definitions proposed in this paper.

The rest of the paper is organized as follows: Section II introduces the SHIWA Simulation Platform and its contribution towards workflow interoperability and meta-workflows. Section III describes the underlying concepts of meta-workflows including atomic and compound tasks, approaches for meta-workflow creation and execution, and different types of meta-workflows. Section IV presents the application of the formal workflow definitions within a public workflow repository when extending its data structure to better accommodate meta-workflows, followed by a description of a scientific use case to demonstrate the effectiveness of the proposed formalism within Astrophysics in Section V. Section VI describes the existing literature related to the work described in this paper. Finally, section VII concludes the paper and lists our future endeavors

II. The SHIWA Simulation Platform

The FP7 SHIWA (Sharing Interoperable Workflows for large-scale scientific simulations on Available DCIs) project [6] created the SHIWA Simulation Platform (SSP) [7] to enable data, infrastructure and workflow interoperability. It supports the whole workflow life-cycle addressing the challenges of (i) workflow interoperability by executing workflows of different workflow systems, (ii) combining workflows of different workflow systems into meta-workflows, and (iii) running workflows on different DCIs and accessing different data storages. SHIWA created the Coarse-Grained Interoperability (CGI) and Fine-Grained Interoperability (FGI) concepts to support workflow interoperability. This paper focuses on the lessons learned from this project with respect to enabling meta-workflows and introduces new concepts and structures based on experiences from this project to improve workflow interoperability.

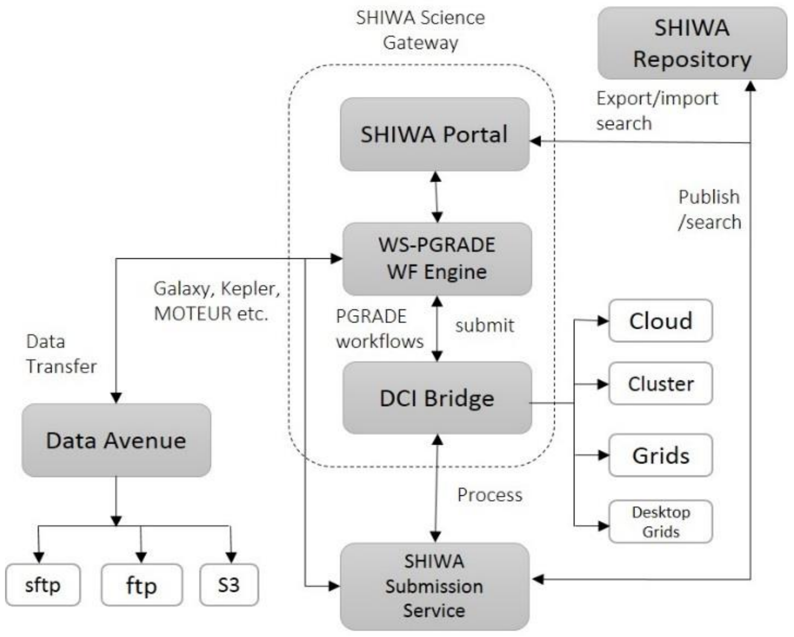


Fig 1: SHIWA Simulation Platform

SHIWA developed the SHIWA Simulation Platform. The simulation platform, presented in Fig 1, contains a science gateway (SHIWA Science Gateway), a submission service (SHIWA Submission Service), and a workflow repository (SHIWA Repository). The SHIWA Science Gateway, built on the gUSE gateway technology [8], incorporates a portal (SHIWA Portal), a workflow system (WS-PGRADE Workflow Management System), and a computing and data resource access service (DCI Bridge and Data Avenue Service). The portal, as the presentation layer of the gateway, contains a graphical workflow editor, a workflow configurator and a workflow execution monitor. The Data Avenue Service and the DCI Bridge represent the infrastructure access layer of the science gateway. The DCI Bridge service provides access to different Distributed Computing Infrastructures (DCI) such as clouds, clusters, desktop and service grids, and supercomputers. Data Avenue Service manages data transfer among different storages using multiple data transfer protocols such as ftp, sftp, S3, etc. The SHIWA Repository is the key service for sharing workflows. It allows publishing and retrieving workflows created using different workflow systems, such as Galaxy, Kepler, MOTEUR, Taverna, WS-PGRADE, etc. These imported workflows can be used by a workflow developer as part of a meta-workflow from within the SHIWA Portal. The repository handles both workflows and workflow engines storing their binaries, configuration, default and meta-data. Developers can publish (or export, or upload), edit, and delete workflows and workflow engines through the repository GUI. Meta-data enables searching workflows based on their application domain. This domain-oriented search can identify the workflows that can be considered for scientific experiments in particular research domains. The meta-data also contains a workflow graph image, a list of input and output ports, and a plain text description of the workflow functionality. The SHIWA Submission Service allows sharing workflows of different workflow systems. First, it imports the workflow from the SHIWA Repository. Next, it either invokes it locally or remotely on pre-deployed workflow engines, or submits workflow engines with the workflow to local or remote resources. The Data Avenue service manages different data formats and data transfer technologies to transfer data used by different workflow systems.

The SHIWA project analyzed major workflow systems and developed a formal description of abstract and concrete workflows and defined the relevant data structures to describe workflows. According to this description, each workflow is represented by a single abstract and one or multiple concrete workflows. We specified the data structure of these workflows in [7]. The abstract workflow specifies the workflow functionality and the workflow graph. The data structure of the abstract workflow contains 5 basic attributes. The *domain* and *tasktype* attributes describe application areas of the abstract workflow and its functionality. They allow straightforward categorization of workflows to support efficient search operations. The *inport* and *outport* attributes define input and output ports of the abstract workflow. The *configuration* attribute specifies parameter requirements of ports. Each abstract workflow may have multiple concrete workflows which represent different implementations of the abstract workflow on the same or in different workflow systems. The concrete workflow defines the implementation of the abstract workflow by specifying the binaries, default input files and parameters needed to run the workflow. Concrete workflows either contain or reference (via URLs) executables required to run the workflow on the associated workflow engine, together with additional meta-data. The key attributes of the concrete workflow are: definition, dependency and configuration attributes. The *definition* attribute is a reference to a file that describes the workflow graph of the concrete workflow. The concrete workflow also stores information about the workflow engine which executes it. The *dependency* attribute handles any requirement of the particular concrete workflow. It can be for instance a DCI middleware, a security infrastructure, or files/executables required for execution. C*onfiguration* attributes resolve these dependencies.

Due to the lack of widespread adoption of meta-workflows at the beginning of the project, the support for meta-workflows in the SHIWA Repository was rather limited. The repository did not allow developers to manage meta-workflows as a set of sub-workflows. It handled the meta-workflow as a monolithic single workflow. This approach led to a few drawbacks. Firstly, it did not allow developers to describe meta-workflows as a set of sub-workflows and customize the sub-workflows to their needs. Secondly, it did not support efficient debugging of meta-workflows. As a meta-workflow can be a complex entity, tracing a fault during execution can be problematic. Identifying sub-workflows for a meta-workflow along with their additional information can facilitate trouble-shooting individual workflows by identifying and locating issues concerning their successful execution.

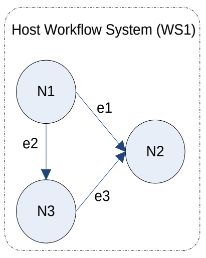
With the emergence and significant rise in the use of meta-workflows within diverse scientific communities, the support for such workflows is paramount to enable workflow sharing across different scientific domains. To address these challenges the original formalism behind the SHIWA Repository has been extended to support the SHIWA and other similar workflow repositories. There were two major challenges to be addressed. First, the data structure must allow managing meta-workflows at both sub- and meta-workflow level, i.e. to be able to specify the sub-workflows of a meta-workflow and provide direct access to these sub-workflows in the repository. Second, it must enable description of how sub-workflows are connected and what kind of data is transferred between them. Furthermore, we envisage it to provide the required impetus to standardize the meta-data required to describe workflows in general and meta-workflows in particular, thereby addressing challenges with respect to workflow interoperability.

III. DEFINITION AND TYPES OF META-WORKFLOWS

Scientific workflows are usually represented using a directed graph where the nodes represent individual tasks or functionalities, whereas the edges represent relationships or dependencies between these tasks. A simple graphical representation of scientific workflows has been presented in Fig 2a where the individual tasks are represented by nodes N1, N2 and N3, and the dependencies between these nodes are represented by edges e1, e2 and e3.

We define the following types of workflows: *single workflow*, *native meta-workflow* and *non-native meta-workflow*. In a *single workflow* all the nodes of workflow are individual jobs that are executed and managed by a single workflow system such as P-GRADE [9], Galaxy [10] or Taverna [11] etc. Fig 2a presents a graphical representation for a single workflow.

We define the term *embedded workflows* to refer to the sub-workflows which constitute a *meta-workflow*. Furthermore, we distinguish meta-workflows based on the workflow engine responsible for the execution of embedded workflows. Within this context, in a *native meta-workflow*, the embedded sub-workflows are all from the same host workflow system (WS1) as demonstrated by Fig 2b. However, in a *non-native meta-workflow*, at least one of the embedded workflows is from external workflow systems, as has been presented in Fig 2c where the two embedded workflows are from workflow systems WS2 and WS3, respectively.



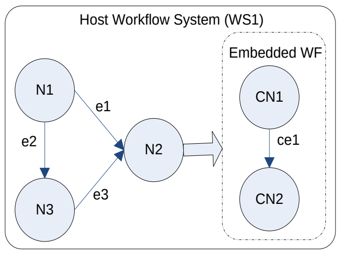


Fig 2a: A single workflow **Fig 2b: Native meta-workflow**

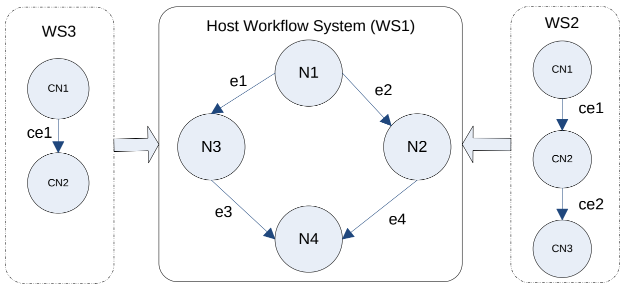


Fig 2c: Non-native meta-workflow

In order to draw a formal representation of the meta-workflow concept, we follow the definitions and guidelines adopted by [7]. A meta-workflow is considered a multi-graph structure represented by <*vertices, edges, source, target*> where the vertices represent jobs, the edges represent dependencies between jobs. Here, the source and target of an edge represents the *dependee* and *depender*, respectively. Firstly, in order to be qualified as a *graph*, a multi-graph structure *G* has to satisfy the following condition:

$$vertices\ (G) \neq \varphi$$

Let *G* represent the meta-workflow graph. *CG1* and *CG2* are two sub-workflow graphs, *G* is a meta-workflow composed of *CG1* and *CG2* if and only if

$$vertices(CG1) \cup vertices(CG2) \subseteq vertices\ (G), \text{ and}$$

$$edges(CG1) \cup edges\ (CG2) \subseteq edges\ (G)$$

A. Approaches for Meta-workflow Creation and Execution

In order to create non-native meta-workflows, two different approaches have been established, i.e. Coarse-Grained Interoperability (CGI) or black box approach, and Fine-Grained Interoperability (FGI) or white box approach [1]. This section explains each of these along with their respective formal representations.

We start with the formal definition of the job and a set of jobs

J =: {J_ABS, J_CNR, J_CNF, J_ENG} (eq. 1)

where J_ABS : abstract description of the job,
J_CNR : implementation or executable of the job,
J_CNF : configuration of the job, and
J_ENG : workflow engine running the job

JOBS = J_ATOM ∪ J_COMP (eq. 2)

where J_ATOM : set of atomic jobs and
J_COMP : set of compound jobs

Next, we borrow the basic definitions of workflows from [7] using the following data models:

WF_ID : the unique workflow ID
J_NA : native job
J_NN : non-native job
WF_NA : set of native workflows
WF_NN : set of non-native workflows
WE_NA : set of native workflow engine(s)
WE_NN : set of non-native workflow engine(s)

workflow:

WF =: {WF_ABS, WF_CNR, WF_CNF, WF_ENG} (eq. 3)

where WF_ABS: abstract workflow
WF_CNR: concrete workflow
WF_CNF: workflow configuration
WF_ENG: workflow engine running the workflow.

abstract workflow:

WF_ABS =: {J_ABS, PORT_IN, PORT_OUT, WF_GRP} (eq. 4)

where J_ABS : abstract description of jobs constituting the workflow
PORT_IN : inputs for the workflow
PORT_OUT : outputs for the workflow
WF_GRP : workflow graph representing the orchestration of jobs

concrete workflow:

J_CNR_k =: {J_ATR_k, J_IMP_k} (eq. 5)

where $J_k \in$ JOBS
J_ATR_k : attributes of the job, such as for instance the environment variables, data locations, command line arguments etc.
J_IMP_k : implementation/executable of the job

WF_CNR =: {J_CNR_1,…, J_CNR_n} (eq. 6)

where J_i : a job such that $J_i \in$ JOBS and i = 1,…, n

meta-workflow:

WF_{meta} = :{J_NA, J_NN, WF_NA, WF_NN}, (eq. 7)

Fine-Grained Interoperability Approach: With FGI, the workflow is treated as having two distinguished parts, i.e. the abstract part and the concrete part. The abstract part includes the abstract input/output functionality of a workflow and the workflow based orchestration of computational tasks. The concrete part contains low level information about the implementation technologies of its computational tasks, such as the method to call a web service or executing an application on a specific machine.

As part of the FGI approach, the abstract part of the workflow is transformed into an *Interoperable Workflow Intermediate Representation (IWIR)* as illustrated by Fig 3. At the abstract level, IWIR is used for describing workflows at a lower level of abstraction that is only processed by the existing workflow systems and not directly exposed to the human developer [1]. The concrete part of the workflow is not transformed but is bundled with the IWIR representation of the abstract part to form an IWIR bundle. The FGI approach for creating meta-workflows has several advantages such as abstraction from high-level workflow language, abstraction from the Distributed Computing Infrastructure (DCI) and the enactment engine, and runtime interoperability among different workflow engines.

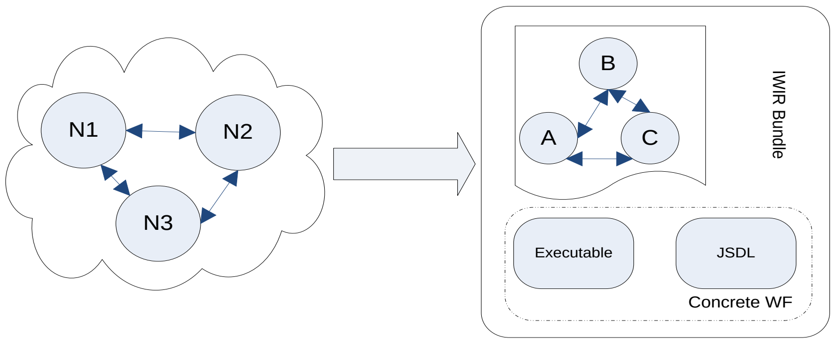


Fig3: Transformation of workflows for FGI approach

Using the structures and definitions presented earlier in this section, we present the formal definition of the FGI approach using Z-notation [12] in Fig. 4. Z is a state-based mathematical specification language including set theory and predicate calculus. The state and operations are conventionally defined and used in a Z specification. The state consists of state variables, and operations are specified upon the state. Our choice of Z notation is motivated by its ease to understand and the flexibility to model a specification which can directly lead to the code. Z notation has been used in contemporary systems to describe a system and its properties such, as [13] and [14].

```
__________________FGI based Meta-workflow Execution______
| ΔMeta_Workflow_Execution_FGI
| WF_ID? : ℕ
| $T_i$ : ATOM_TASKS
| $T_j$ : COMP_TASKS
| Port_i : PORT_IN
| Port_j : PORT_OUT
| WF = {WF_ABS, WF_CNR, WF_CNF, WF_ENG}
|_______________________________________________________
| WF_ID > 0
| PORTS = PORT_IN ∪ PORT_OUT
| WF_ABS =: {J_ABS, PORT_IN, PORT_OUT, WF_GRP}
| JOBS = J_ATOM ∪ J_COMP
| $J_k$ =: {$J\_ABS_k$ , $J\_ATR_k$, $J\_IMP_k$}
|
| Iff $J_k$ ∈ JOBS
|     WF_CNR =: {$J\_ATR_k$, $J\_IMP_k$}
|     IWIR_WF_G->Create_abs_IWIR_WF(WF_ABS)
|
| Foreach $J_x$ ∈ JOBS ⇸WF_ABS
|     $J\_CNR_x$ : DCI_Ind_CNR($J\_IMP_x$)
|     JOBS_CNR = {$J\_CNR_x$}
|     IWR_Bundle = Bundle(IWIR_WF, JOBS_CNR)
|________________________________________________________
```

Fig 4: Formal definition of the FGI approach using Z-notation

Coarse-Grained Interoperability Approach (CGI): It is focused at defining a workflow engine based interoperability concept independent of the workflow systems, allowing sharing workflows from different workflow systems [7]. In this approach the non-native workflows and workflow engines are managed as legacy applications by the native workflow system.

Fig 5 presents a graphical representation of the CGI approach for workflow interoperability exemplifying the sequence of events to execute a non-native workflow using a repository and a submission service. The formalization for the CGI approach is presented in Fig 6 which extends on our initial effort in [15] and makes use of the Z-notation. Further details of the CGI approach and method used to achieve formalism are elaborated in [5].

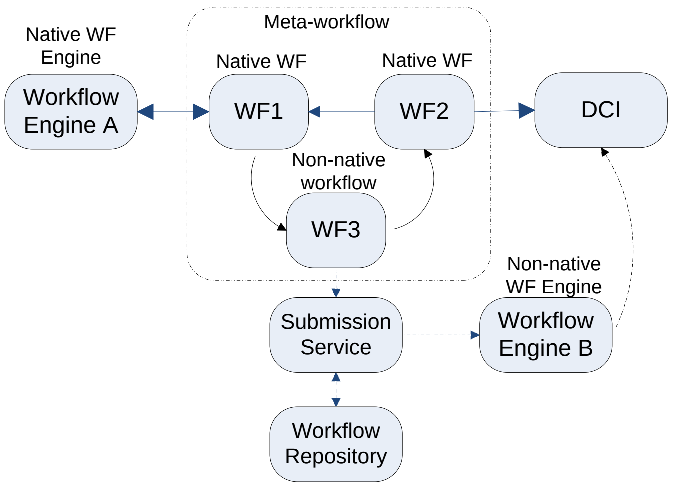


Fig 5: Coarse-grained workflow interoperability usage scenario

```
________CGI based Meta-workflow Execution____________
| $\Delta$ Meta_Workflow_Execution_CGI
| WF_ID? : $\mathbb{N}$
| $wf_{na}$?: WF_NA
| $wf_{nn}$?: WF_NN
| $we_{na}$?: WE_NA
| $we_{nn}$?:WE_NN
|______________________________________________
| WF_ID > 0
| WF_NA $\neq$ <>
| WF_NN $\neq$ <>
| WE_NA $\neq$ <>
| WE_NN $\neq$ <>
|
| If $wf_i \in$ WF_NN
|         $wf_{nn} = wf_i$
|         $we_{nn} = wf_i (WF_{eng})$
|         Execute_Submission_Service($wf_{nn}$ , $we_{nn}$ )
| Else
|         $wf_n = wf_i$
|         $we\_n_n$ = wfi(WFeng)
|         Execute_Native_WFEngine($wf_n$ , $we\_n_n$)
|______________________________________________
```

Fig 6: Formal definition of the CGI approach using Z-notation

Although we introduced the concepts surrounding both CGI and FGI approaches to meta-workflows, the focus of this paper will be limited to CGI based meta-workflows due to their support in the SHIWA Simulation Platform.

B. *Types of CGI-based Meta-workflows*

In this section we describe the different types of meta-workflows along with their formal definitions. These definitions are envisioned to improve the understanding of the attributes and semantics of each type of meta-workflows thereby facilitating workflow developers to design new workflows.

Single job meta-workflow: This type of meta-workflow is a special meta-workflow that represents a workflow with one job in the native workflow system. The job representing this workflow can be a simple native job, a native workflow or even a non-native workflow. Fig. 7 presents a graphical representation of this workflow type. As we have emphasized as part of our formalization in [7], jobs can be native and non-native.

As a job can be *native* or *non-native* depending upon its workflow engine, we define workflow engine of a job as:

J_ENG =: {ENG_NA, ENG_NN} (eq. 8)

where ENG_NA : native workflow engine

ENG_NN : non-native workflow engine

Therefore the single job meta-workflow can be described using eq. 3 and eq. 7 such as

WF_{snj} =: {J_1} = : {J_ABS, J_CNR, J_CNF, J_ENG} (eq. 9)

where J_1 : {$J_{na} \cup J_{nn} \cup WF_{na} \cup WF_{nn}$},

J_{na} : a native job,

J_{nn} : a non-native job,

WF_{na} : a native workflow, and

W_{nn} : a non-native workflow.

Linear multi-job meta-workflow: This is a pipeline of multiple jobs in the native workflow system where any (or even all) of these jobs can be non-native workflows. The execution of each job depends on the receipt of input files from previous jobs. Fig. 8 presents a graphical representation of this type of workflow. As linear multi-job meta-workflows are composed of single jobs and/or single job meta-workflows, we use our formalization from previous sections to describe this type of meta-workflow. Therefore, a linear multi-job meta-workflow can be described using our basic definition of meta-workflows from eq. 7 such that

WF_{lmj} =: {J_1,..,J_m}= :{J_NA, J_NN, WF_NA, WF_NN} (eq. 10)

where J_j : job of linear multi-job workflow, j = 1,.., m. and

J_j {$J_{na} \cup J_{nn} \cup WF_{na} \cup WF_{nn}$}

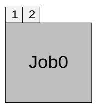


Fig. 7: Single job meta-workflow

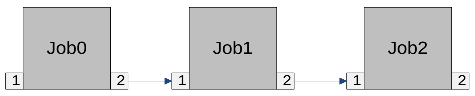


Fig. 8: Linear multi-job meta-workflow

Now, in order to formalize a linear multi-job meta-workflow, let us define an edge as a connection between an input and an output port. By definition, an edge is defined as *e = (i,o)* where *i* is the input port and *o* is the output port. Consequently if P_i is the input port and P_o is the output port of job *J*, then edge is defined as

e = (P_i, P_o).

Therefore, for a linear multi-job meta-workflow, if J_x and J_y are any two consecutive jobs and J_x is not the last job of the workflow, there exists a valid edge such that

e_{xy}= (P_{yi}, P_{xo}) (eq. 11)

where P_{yi} : the input port of J_y.

P_{xo} : the output port of J_x and

Parallel multi-job meta-workflow: This is a workflow that includes parallel branches. One or more of these branches can include one or more embedded non-native workflows. Parallel multi-job meta-workflows are composed of several linear multi-job meta-workflows as presented in Fig. 9.

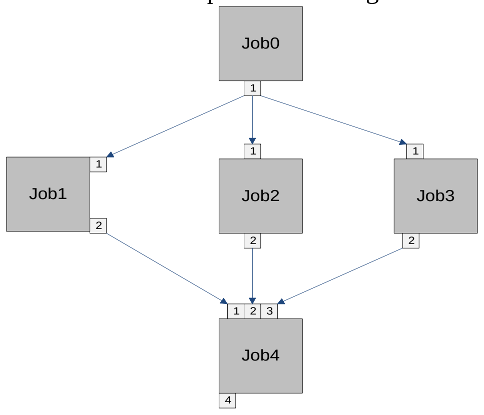


Fig. 9: Parallel multi-job meta-workflow

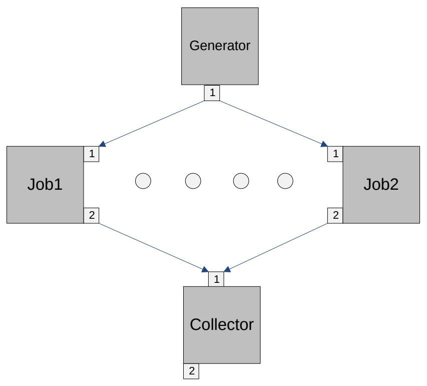


Fig. 10: Parameter sweep meta-workflow

To formalize parallel multi-job meta-workflows we have to define connections between their jobs. Let J_s represent a split job (a job that is followed by several parallel worker jobs), and J_{wx} represent a worker job where x = 1…k. Here, a *worker job* refers to a job assigned with a specific task within the workflow. A parallel multi-job meta-workflow must contain a “split” edge.

$e_{swx} = (P_{so}, P_{wxi})$ (eq. 12)

where P_{so} : the output port of the split job J_s

P_{wxi} : the input port of the worker job J_{wx}

Furthermore, let J_m represent the merge job that merges the results of parallel workers jobs. Therefore, a parallel multi-job meta-workflow must also have a merge edge e_{wxm} such that

$e_{wxm} = (P_{wxo}, P_{mix})$ (eq. 13)

where P_{wxo} : the output port of a worker job J_{wx}, and

P_{mix} : the input port of the merge job J_m.

Utilizing formalization from eq. 7, the parallel multi-job meta-workflow can be described as

$WF_{pmj} =: \{J_1,\ldots,J_n\} =: \{ J_s, J_{w1},\ldots J_k, J_m\} = :\{J\_NA, J\_NN, WF\_NA, WF\_NN\}$ (eq. 14)

where J_j job of a parallel multi-job meta-workflow $j = 1\ldots n$,

J_s : {J_NA, J_NN, WF_NA, WF_NN},

J_{wx} : {J_NA, J_NN, WF_NA, WF_NN}, and

J_m : {J_NA, J_NN, WF_NA, WF_NN},

Parameter sweep meta-workflow: This represents a parameter sweep workflow that contains a generator job, several worker jobs, and a collector job. The generator job produces a number of inputs each to be consumed by a worker job. Here, a *worker job* refers to job assigned with a specific task within the workflow. The collector job aggregates the outputs of all the worker jobs and prepares the final output. Although this functionality is very similar to the parallel multi-job meta-workflow, the primary difference between the two is that the worker jobs for parameter sweep workflow are generated dynamically and are not known at the configuration stage of the workflow. Therefore the relations between the generator, worker and collector jobs are dynamic as compared to the parallel multi-job workflow, as presented in Fig. 10. Also, while workers in parallel multi-job meta-workflows can represent different functionality, in case of parameter sweep meta-workflows all workers represent the same functionality with different parameter values only. The connections between the generator, collector and worker jobs for a parameter sweep meta-workflow can be described as:

$e_{gwx} = (P_{go}, P_{wyi})$ (eq. 15)

where P_{go} : the output port of the generator job J_g

P_{wyi} : the input port of the worker job J_{wy} and $y = 1 \ldots l$

Furthermore, let J_c represent the collector job A parameter sweep meta-workflow must also have an edge e_{wyc} such that

$e_{wyc} = (P_{wyo}, P_{ci})$ (eq. 16)

where P_{wyo} : the output of a worker job J_{wy}, and

P_{ci} : an input port of collector job J_c.

Due to the similarity with the parallel multi-job meta-workflow, parameter sweep meta-workflow can be described using eq. 7 as:

$WF_{psj} =: \{J_1,\ldots,J_n\} =:\{ J_g, J_{w1},\ldots J_l, J_c\}= :\{J\_NA, J\_NN, WF\_NA, WF\_NN\}$ (eq. 17)

where J_j : job of parameter sweep meta-workflow, $j =1,\ldots,m$

J_g : {J_NA, J_NN, WF_NA, WF_NN},

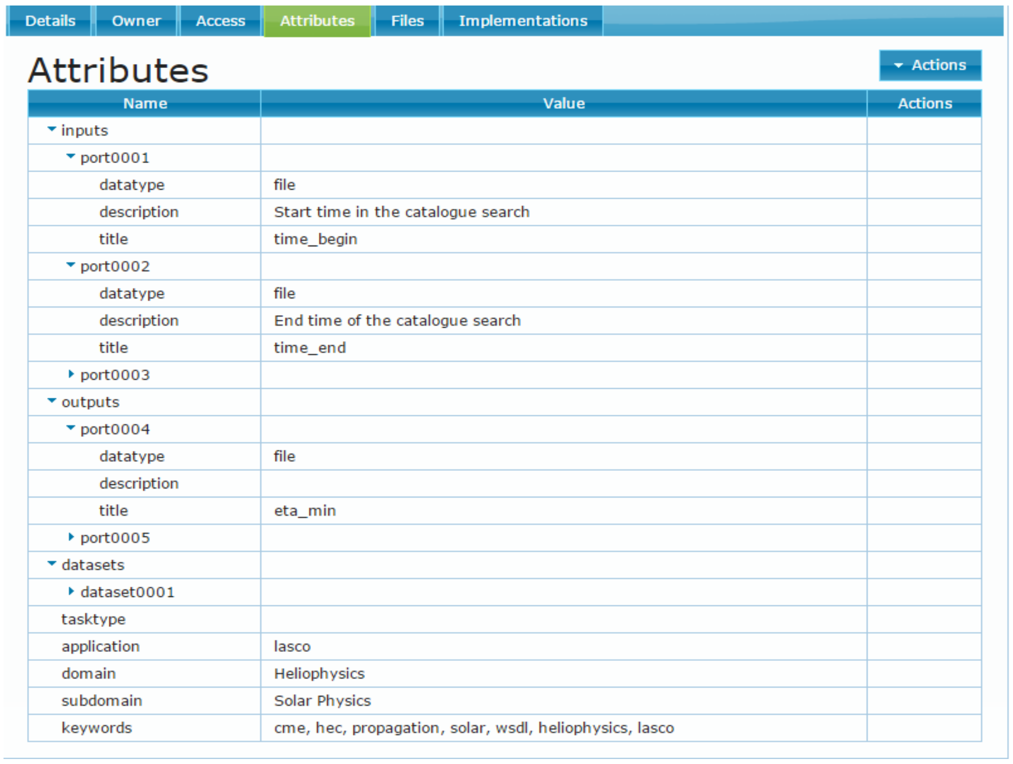
J_{wy} : {J_NA, J_NN, WF_NA, WF_NN}, and

J_c : {J_NA, J_NN, WF_NA, WF_NN},

IV. Metaworkflow Definitions for Workflow Repositiories

In order to illustrate the potential application of the formal meta-workflow definitions introduced in section III, this section describes how the data structure of a workflow repository can be improved to better accommodate meta-workflows. As part of the SHIWA project, the SHIWA Repository has been developed to support the sharing of workflows across different scientific domains. However, due to the lack of widespread adoption of meta-workflows at the beginning of the project, the support for such meta-workflows in the repository is rather limited. The current generic data structure for a workflow in the SHIWA Repository is described in [7], whereas Fig 11 displays the information about an example meta-workflow published in the repository.

As it can be seen in the figure, information available for the meta-workflow in the repository is limited to input and output port definitions (*inputs/port0001, inputs/port0002, inputs/port0003, and outputs/port0004*), sample datasets (*datasets/dataset0001*), and generic metadata (e.g. *application, domain, subdomain, keywords*). Therefore, the repository describes the meta-workflow as a complete black-box, not including any information about its sub-workflows. This is especially disadvantageous in that it does not allow the user of such workflow to identify the sub-workflows and if possible customize them to their needs. For instance, in case of FGI and template based approaches for meta-workflow creation, a user is able to customize the sub-workflows by making modifications allowed by the workflow developer. However, with the current implementation of data structures within the SHIWA Repository, it is not possible. Another use-case requiring such detailed information about sub-workflows is to facilitate the efficient debugging of meta-workflows. As a meta-workflow can be a complex entity, tracing a fault in its successful execution can be problematic. Identifying sub-workflows for a meta-workflow along with their additional information can facilitate trouble-shooting individual workflows by identifying and locating issues concerning their successful execution.



Attributes

Name	Value	Actions
inputs		
port0001		
datatype	file	
description	Start time in the catalogue search	
title	time_begin	
port0002		
datatype	file	
description	End time of the catalogue search	
title	time_end	
port0003		
outputs		
port0004		
datatype	file	
description		
title	eta_min	
port0005		
datasets		
dataset0001		
tasktype		
application	lasco	
domain	Heliophysics	
subdomain	Solar Physics	
keywords	cme, hec, propagation, solar, wsdl, heliophysics, lasco	

Fig 11: Information stored for a meta-workflow in the SHIWA Repository

element	sub-element	Description	data type
Domain	sub-domain	application domain/sub-domain	List
Tasktype		type of task the workflow executes	plain text
workflow type		simple or meta-workflow	
sub-workflow		sub-workflows in case of workflow_type = meta-workflow	
	workflow_uid	UID of the workflow	
	description		
Inputs		workflow input port	
	number		List
	description		plain text
	data type		List
	sub-workflow mapping	sub-workflow to be mapped to the port	
Outputs		workflow output port	
	number		List
	description		plain text
	data type		List
	sub-workflow mapping	sub-workflow to be mapped to the port	
configuration		resolution of in- & output ports data requirements	
	Portref	input port referenced	plain text
	Value	value passed to the port	plain text
Keyword		domain/sub-domain keywords	List

Table1: Data structure for meta-workflow in the **SHIWA Repository**

Remark: The new elements of the data structure are highlighted in grey in Table 1.

With the emergence and significant rise in the use of meta-workflows within diverse scientific communities, the support for such workflows is paramount to enable workflow sharing across different scientific domains. The formalism proposed as part of this paper will facilitate achieving this above objective by extending the data structures supported by the SHIWA and other similar workflow repositories. Furthermore, we envisage it to provide the required impetus to standardize the meta-data required to describe workflows in general and meta-workflows in particular thereby addressing challenges with respect to workflow sharing and interoperability.

The Table 1 presents the proposed data structure for workflow repositories based on the formalized description presented in this paper to accommodate meta-workflows. We have included the *sub-workflow* element to represent the structure containing a set of sub-workflows embedded in the meta-workflow. This structure is envisaged to provide linkages to sub-workflows present in the repository and therefore will facilitate a user to explore further details of each sub-workflow. We have also included the sub-element *sub-workflow mapping* for both the input and output ports. This is to provide the detail of how the different sub-workflows are orchestrated within the meta-workflow. We envision continuing this research by investigating implementation of this approach within the SHIWA Repository to improve the capabilities offered by the SHIWA Repository for sharing of meta-workflows.

V. THE SCIENTIFIC USE CASE

The results of the research presented in this paper have been adopted by diverse scientific disciplines. We have presented the successful scientific use-case from the Heliophysics community in [5]. In this section, we describe the experience of the Astrophysics community in mapping their use case to the formal descriptions described earlier in this paper. The scientific details and significance of this use case have been presented in [4] and have been further elaborated as part of the ER-Flow project deliverables [16].

Hubble Space Telescope (HST) use case

The International Astronomical Virtual Observatory (IVOA) produces unified virtual data, and offers services to perform complex data discovery and processing tasks across the whole range of astronomical resources provided by Astronomical Data Centers all around the world. IVOA developed a set of standards and services that are commonly and widely used by Astronomers to access archives and catalogues, and to perform operations on observed data.

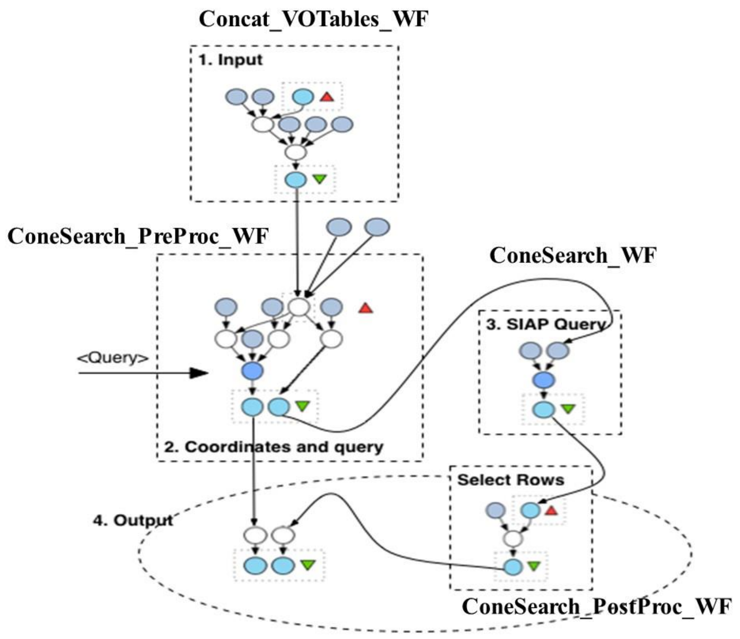


Fig 13: Hubble Space Telescope data access and manipulation meta-workflow

In order to facilitate access to the IVOA resources and services, a large number of data-oriented workflows have been developed [4]. These workflows, built on the AstroTAVERNA plugin [17] of the TAVERNA workflow engine, are data-oriented workflows that are used to access and manipulate astronomical data. They deliver atomic operations that can be reused as basic components (sub-workflows) in other complex astronomical meta-workflows. We created the IVOA workflow library importing the AstroTaverna Virtual Observatory workflows into the SHIWA Repository as abstract workflows. It allows workflow developers designing and implementing complex astronomical scientific experiments. These experiments combine data access based on IVOA standards and data processing operations (e.g. data reduction and analysis). This library has been widely used in the STARNet gateway federation [18] to build astronomical applications. STARNet is a federated network of science gateways based on the gUSE science gateway technologies. It is explicitly designed and tuned to the needs of the astronomical and astrophysical (A&A) community in Europe. This federated gateway infrastructure shares a common authentication system, a distributed computing infrastructure, data archives, portlets, and workflow repositories allowing scientists to explore new collaboration opportunities and advancing the scientific research activity within A&A. Building upon these technologies, a number of challenging applications from different A&A domains have been successfully prototyped and tested including [4, 16, 18].

The Hubble Space Telescope (HST) use case will be presented in the rest of this section to outline how to create meta-workflows. The HST archive offers access to the HST observations using the IVOA services. Data can be retrieved in the form of FITS files or VOTables (XML) files [19]. The Hubble Space Telescope data access and manipulation meta-workflow incorporates workflows from the IVOA workflow library. The meta-workflow has four phases each one of these is implemented by a sub-workflow available in the IVOA workflow library, as shown in Fig. 13:

1. **Concatenating VOTables workflow**: Concat_VOTables_WF
   One of the common astronomical data processing tasks is to combine data retrieved from archives or catalogues. This workflow selects astronomical objects to be searched in the HST catalogue using their source name from an ASCII list. The inputs of the workflow, presented in Fig. 14, are VOTables. This workflow combines them into a vertically replicated single table. In the search operation source names in different astronomical catalogues could be different.

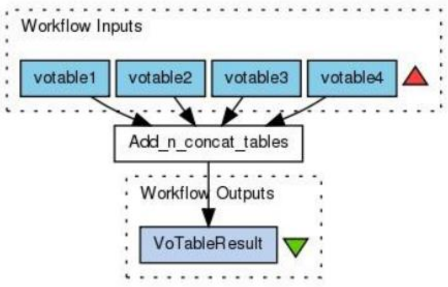


Fig 14: Concatenating VOTables workflow

2. **Cone search pre-processing workflow**: ConeSearch_PreProc_WF
   First, it converts source names into astronomical coordinates to allow cone search [20]. Next, it generates an XML file (VOTable) with a list of cone searches to be executed on the HST SIAP image service (Simple Image Access Protocol) [21].

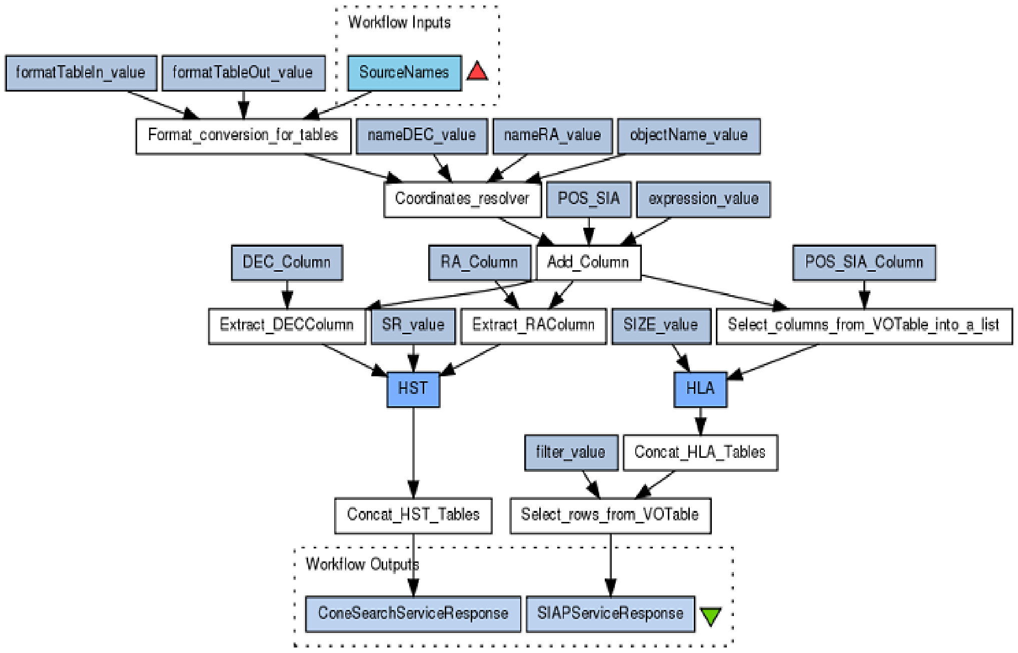


Fig 15: HST Cone Search Workflow

3. **Cone search workflow**: ConeSearch_WF
   It queries the HST SIAP service using the list of cone searches compiled in the previous phase. The cone search workflow, presented in Fig 15, runs a cone-search around a point in the Sky. The input of the workflow is an ASCII file with a list of source names. First, this file is converted into a VOTable and an extra column is added for the coordinates that are necessary to query the SIAP image service. Next, the HST SIAP service is queried. The final result consists of two different VOTables called Concat_HST_Tables. The output of this workflow can be processed and further analysed. For example it is possible to make photometric analysis using the SExtractor workflow (Source Extractor) [17], stored in the IVOA workflow library, to find all sources from the HST images and evaluating their magnitude and position.
4. **Cone search post-processing workflow**: ConeSearch_PostProc_WF
   The final query result is processed to obtain two different VOTables created by the VO Services.

Since the above listed workflows are all Taverna workflows, they have to be wrapped in WS-PGRADE workflows to enable their execution on the SHIWA Simulation Platform using the Coarse-Grained Interoperability approach, as presented in Section III. Having wrapped Taverna workflows, they can be executed on their own or can be added to meta-workflows.

Following our definitions in section II, the above mentioned workflows are *non-native.* Therefore, the HST meta-workflow can be written as:

HST_metaWF =: {J_NA, J_NN, WF_NA, WF_NN}
where
WF_NN = {Concat_VOTables_WF, ConeSearch_WF, ConeSearch_PreProc_WF, ConeSearch_PostProc_WF}

Using the formal model for CGI based meta-workflow execution in section III,

HST_Metaworkflow_Execution

Δ *HST_metaWorkflow_Execution_CGI*
WF_ID? : $\mathbb{N}$
wf_{nn}*?: WF_NN*
we_{nn}*?:WE_NN*

WF_ID > 0
WF_NN = {Concat_VOTables_WF, ConeSearch_WF, ConeSearch_PreProc_WF, ConeSearch_PostProc_WF}
WE_NN = {Taverna}

$wf_{nn} = wf_i$
$we_{nn} = wf_i$ *(WF_ENG)*
Execute_Submission_Service(wf_{nn} , we_{nn})

Fig 16: Formal definition of HST workflows using CGI approach

Within the context of the meta-workflow definitions presented earlier in this paper, The Hubble Space Telescope (HST) data access and manipulation meta-flow is a linear multi-job meta-workflow that can be described by eq. 10 and eq. 11.

VI. RELATED WORK

The usage of scientific workflow paradigm has been widely adopted to address problems across widespread domains such as Computational Chemistry [22] [23], [24], Astrophysics [25], or Bioinformatics [26]. In order to facilitate workflow development, various workflow systems have been implemented which enable the process of workflow creation, configuration and execution. Examples of such workflow systems include Taverna [11], MOTEUR[27], Galaxy [10], and P-GRADE [9].

As many different workflow systems have been developed based on different programming models having different internal workflow description, interoperability across workflows generated by different workflow systems is a non-trivial challenge. Workflow interoperability is fundamental to facilitate sharing of workflows across different workflow systems and therefore has attracted significant attention from the scientific community. Within this context, four major approaches for workflow interoperability include *workflow language standardization*, *workflow translation*, *workflow engine integration,* and *sharing among data sources* [7].

A number of recent efforts focus on addressing the workflow interoperability problem, establishing mechanisms to facilitate sharing of scientific workflows, and creating meta-workflows.

The SHIWA (Sharing Interoperable Workflows for large-scale scientific simulations on Available DCIs) [6] project focused at addressing the challenge of workflow interoperability by using the Coarse-Grained Interoperability (CGI) and Fine-Grained Interoperability (FGI) concepts. The CGI concept is based on workflow engine integration, embedding and nesting workflows of different workflow systems [7]. The FGI concept is built on workflow language translation using the Interoperable Workflow Intermediate Representation (IWIR) for common workflow description [1]. The SHIWA project has developed a workflow repository, the SHIWA Workflow Repository, which enables publishing and retrieving workflows created using different workflow systems such as Galaxy, Kepler, MOTEUR, Taverna, WS-PGRADE, etc. These imported workflows can be used by a workflow developer as part of a meta-workflow from within the SHIWA Portal.

Furthermore, there has been a significant rise in the interest from diverse scientific communities with regards to the use of meta-workflows to facilitate achieving complex experimentation. The authors in [3] present an effort from the Heliophysics community detailing their experiences with regards to using meta-workflows to "re-think and re-design with the aim of fostering re-usability" and to aid usability with respect to workflow execution. The authors make use of multiple workflow engines, i.e. Taverna and WS-PGRADE, to develop atomic workflows which are re-used as building blocks by users to compose more complex meta-workflows to address specific science cases. The case study presented in the previous section is an example of such meta-workflows. These experiences are significant in that they represent an effort to compose meta-workflows using both native and non-native workflows thereby highlighting the vast possibilities of meta-workflows.

Moreover, [22], [23] and [24] represent efforts from Computational Chemistry discipline to use meta-workflows with the objective to enhance re-usability across different methodologies in Computational Chemistry, namely molecular dynamics and quantum chemistry. The authors present their experiences with the WS-PGRADE technology to develop meta-workflows by using atomic workflows from within the Quantum Chemistry domain and by re-using basic workflows developed for the molecular dynamics domain.

The authors in [28] and [29] represent efforts from Neuroimaging to use meta-workflows to achieve complex data processing and analysis using high performance computing infrastructures. The overall objective of using meta-workflows in this case is to foster collaboration among different research groups by sharing data processing applications, data and workflows. Both of these efforts highlight the use of WS-PGRADE tools developed as part of the SHIWA project to achieve their objectives. These efforts demonstrate the effectiveness of the meta-workflow concepts to facilitate workflow interoperability and sharing across diverse scientific domain by using different workflow technologies.

The ClowdFlows [30] project aims to facilitate workflow creation, execution and sharing using a user friendly web based front end. It allows the users to construct new workflows using elements called *widgets* and also enables them to share their workflows via a workflow repository. The platform also facilitates the creation of meta-workflows. Abouelhoda et al. [26] present another approach to facilitate creation and execution of meta-workflows with specific emphasis on the bioinformatics scientific community. The approach focuses on two workflow engines i.e. Taverna and Galaxy which are widely used within the bioinformatics community and develops a software system called *Tavaxy* to combine advantages of both systems. Wings Project [31] is an extension to the Pegasus workflow engine with special emphasis on execution and management of very large scientific workflows. The approach exploits semantic representation of workflows to create workflow templates which can then be used to create workflows or meta-workflows of any scale.

In addition to the above described literature, there have been considerable efforts with respect to formalism for workflows. In this respect, a number of efforts have utilized petri nets to achieve this due to multifold reasons, as explained in [32]. Furthermore, workflow schemas have also been used to represent formal foundations of workflows. For instance, [8] presents an effort to achieve improved flexibility of the workflow management systems by using workflow schemas. The authors propose to use workflow schemas to represent different workflows with the aim to enable them to dynamically change the structure of the workflow instances.

The authors in [33] introduce an effort for formal semantics of workflows for the Taverna2 workflow management system. The primary objective of this formalism is to define "when two workflow specifications are equivalent and to allow reasoning about what can and cannot be expressed in Taverna". The calculus presented includes formal representation of the workflows as well as the traces of events within the workflow execution. The formalism presented by the authors is complex and focused at representing individual workflows and the respective operations with the overall aim to determine if two given workflows are identical.

Within the context of interoperability, [7] represent an effort to provide formal foundation for workflows to facilitate workflow sharing and interoperability. The authors propose fundamental definitions of workflows with identification of different components of a workflow such as the *abstract* and *concrete* parts of a workflow. This formalism envisioned to enable improved representation of the workflows within the SHIWA simulation platform in general and to enable creation of meta-workflows in particular. The formal notations proposed in this paper have extended the definitions proposed by [7] to achieve sharing of workflows and creation of meta-workflows.

Although meta-workflows have been introduced in various related papers, no attempt has been made to categorize and systematically describe meta-workflow types and their design patterns.

VII. CONCLUSIONS

The paper has contributed towards the definition and analysis of meta-workflow approaches for complex scientific meta-workflows. It has presented the formal definitions of different types of meta-workflows including formalizing the CGI-based and FGI-based meta-workflow execution approaches. The paper has also identified the need for a standard to describe scientific workflows to facilitate workflow sharing. Furthermore, it has demonstrated the use of formal notations presented in this paper to describe meta-workflows for the SHIWA repository. The paper has also presented experimentation for Astrophysics as a proof of concept for the approach presented in the paper. The authors plan to undertake the implementation of proposed formalism for workflow sharing as part of a workflow repository in near future.

VIII. ACKNOWLEDGEMENTS

We would like to acknowledge the contribution of Dr. Eoin Carley (TCD-Physics) and Dr. Claudio Vuerli (INAF-OATS) for this research.

REFERENCES

[1] Plankensteiner K., Prodan, R., Janetschek, M., Fahringer, T., Montagnat, J., Rogers, D., Harvey, I., Taylor, I., Balasko, A., Kacsuk, P.; Fine-Grained Interoperability Of Scientific Workflows In Distributed Computing Infrastructures, In The *Journal Of Grid Computing*, Vol 11(3), 429-455, 2013

[2] Mosgrid Science Gateway, https://www.mosgrid.de/ [Last Accessed: 29th June 2016]

[3] Pierantoni, G. And Carley, E.; Metaworkflows And Workflow Interoperability For Heliophysics, In The Proceedings Of The *6th International Workshop On Science Gateways*, 79-84, 2014

[4] Castelli, G., Taffoni, G., Sciacca, E., Becciani, U., Costa, A., Krokos, M., Pasian, F., Vuerli, C.; VO-Compliant Workflows And Science Gateways, 2015, *Astronomy And Computing*, Volume 11, 102-108.

[5] Arshad, J., Terstyanszky, G., Kiss, T., Weingarten, N.; A Definition And Analysis Of The Role Of Meta-Workflows In Workflow Interoperability, In the Proceedings of the *7th International Workshop On Science Gateways (IWSG)* 8-15 Budapest, Hungary, 2015

[6] SHIWA: Sharing Interoperable Workflows For Large-Scale Scientific Simulation On Available DCIs. *http://www.shiwa-workflow.eu*, 2011. [Last Acccessed: 29th June 2016]

[7] Terstyanszky, G., Kukla, T., Kiss, T., Kacuuk, P., Balasko, A., Farkas, Z.; Enabling Scientific Workflow Sharing Through Coarse Grained Interoperability in *the Future Generation Computer Systems* Vol 37, 46-59, 2014.

[8] Guse Grid And Cloud Science Gateway Available online At: *https://sourceforge.net/projects/guse/* [Last Accessed: 21st June 2016]

[9] Kertész, A., Sipos, G., And Kacsuk, P.; Brokering Multi-Grid Workflows In The P-GRADE Portal, in *Euro-Par 2006*: *Parallel Processing*, Vol. 4375, Springer, Berlin, 138–149, 2007.

[10] Giardine, B., Riemer, C., Hardison, R. C., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., Albert, I., Taylor, J., Miller, W., Kent, W. J., Nekrutenko, A..; Galaxy: A Platform For Interactive Large-Scale Genome Analysis. *Genome Research*, 15(10):1451-5, 2005.

[11] Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M. R., Wipat, A., Li, P.;Taverna: A Tool For the Composition and Enactment of Bioinformatics Workflows. *Bioinformatics*, 20(17):3045-54, 2004

[12] ISO/IEC 13568, Information Technology- Z Formal Specification Notation- Syntax, Type System And Semantics, *International Standard*, 2002

[13] Woodcock, J., Stepney, S., Cooper, D., Clark, J., And Jacob, I., The Certificationof The Mondex Electronic Purse To ITSEC Level E6, *Formal Aspects Of Computing* 20(1) 2008

[14] Mcdermott, And Freitas, L.; A Formal Security Policy For Xenon, *ACM Conference on Computer And Communications Security*, Virginia, USA, 43-52, 2008

[15] Toda, Y.; The Formalization Of Simple Graphs, *Formalized Mathematics*, Vol.5, No. 1, 137-144, 1996

[16] Virtual Observatory Use-Case Report. ER Flow Project. Available At: http://www.erflow.eu/virtual-observatory-use-case [Last Accessed: 29th June 2016]

[17] Bertin, E. & Arnouts, S; Sextractor: Software For Source Extraction, *Astronomy & Astrophysics Supplement* 317, 1996, 393

[18] Starnet Gateway Federation At: *www.oact.inaf.it/starnet/* [Last Accessed: 29th June 2016]

[19] Mink, J.; Mann, R. G.; Hanisch, R.; Rots, A.; Seaman, R.; Jenness, T.; Thomas, B.; O'Mullane, W. The Past, Present, And Future Of Astronomical Data Formats, *2015ASPC*, 495, 11M

[20] Williams, R., Hanisch, R., Szalay, A., Plante, R.; IVOA Recommendation: Simple Cone Search Version 1.03, *Astro-Ph.IM*, Arxiv:1110.0498

[21] Dowler, P., Tody, D., Bonnarel, F.; IVOA Simple Image Access, 2016, *Astro-Ph.IM*, Arxiv:1601.00519

[22] Herres-Pawlis, S., Hoffman, A., Balasko, A., Kacsuk, P., Birkenheuer, G., Brinkman, A., Garza, L., Kruger, J., Gesing, S., Grunzke, R., Terstyansky, G., Weingarten, N..; Quantum Chemical Meta-Workflows In Mosgrid in the *Concurrency and Computation: Practice And Experience* 27(2), 344-357, 2015.

[23] Herres-Pawlis, S., Hoffman, A., Garza, L., Kruger, J., Gesing, S., Grunzke, R., Nagel, W. E., Terstyansky, G., Weingarten, N.; Meta-Metaworkflows For Combining Quantum Chemistry And Molecular Dynamics In The Mosgrid Science Gateway in *the 6th International Workshop On Science Gateways.* 73-78. 2014
[24] Herres-Pawlis, S., ., Hoffman, A., Gesing, S., Kruger, J., Balasko, A., Kacsuk, P., Grunzke, R., Birkenheuer, G., Packschies, L. ; User-Friendly Metaworkflows In Quantum Chemistry, *In The IEEE International Conference On Cluster Computing*, 1-3, 2013.
[25] Becciani, U., Sciacca, E., Costa, A., Massimino, P., Pistagna, C., Riggi, S., Vitello, F., Petta, C., Bandieramonte, M., Krokos, M.; Science Gateway Technologies For The Astrophysics Community, *in the Concurrency And Computation: Practice And Experience*, 2014
[26] Abouelhoda, M., Alaa, S., Ghanem, M.; Meta-Workflows: Pattern-Based Interoperability Between Galaxy And Taverna In The *International Workshop On Workflow Approaches For New Data-Centric Science*, 2010.
[27] Glatard T., Montagnat J., Lingrand D, Pennec X.; Flexible And Efficient Workflow Deployment Of Data-Intensive Applications On Grids With MOTEUR, *In The International Journal Of High Performance Computing Applications*, 2008
[28] Korkhov, V., Krefting D, Montagnat J, Truong Huu T, Kukla T, Terstyanszky G, Manset D, Caan M, Olabarriaga S.; SHIWA Workflow Interoperability Solutions For Neuroimaging Data Analysis, *In The Studies In Health Technology And Informaitcs* 175:109-10, September 2012.
[29] Korkhov, V., Krefting, D., Kukla, T., Terstyanszky, G., Caan, M., Olabarriaga, S. D.; Exploring Workflow Interoperability Tools For Neuroimaging Data Analysis, *In The Proceedings Of The 6th Workshop On The Workflows In Support Of Large-Scale Science*, Seattle, U.S. 87-96 2011.
[30] Kranjc, J., Podpecan, V., Lavrac, N.; Clowdflows: A Cloud Based Scientific Platform, *In Machine Learning And Knowledge Discovery In Databases*, LNCS, Vo.. 7524, 816-819, 2012
[31] Gil, Y., Ratnakar, V., Dellman, E.; Wings For Pegasus: A Semantic Approach To Creating Very Large Scientific Workflows, *In OWLED 2006*.
[32] Van Der Aalst, W. M. P.; The Application Of Petri Nets To Workflow Management, *Journal Of Circuits Systems Comput. 8* (1) (1998) 21–66.
[33] Sroka, J., Hidders, J., Missier, P., Globe, C.; A Formal Semantics For The Taverna2 Workflow Model, 2010, *Journal Of Computer And System Sciences*, Volume 76, P. 490-508