A Definition and Analysis of the Role of Meta-workflows in Workflow Interoperability

Junaid Arshad, Gabor Terstyanszky, Tamas Kiss, Noam Weingarten
Center for Parallel Computing, University of Westminster, London, UK
(J.Arshad, G.Z.Terstyanszky, T.Kiss, Weingan)@westminster.ac.uk

***Abstract*—Scientific workflows orchestrate the execution of complex experiments on high performance computing platforms. Meta-workflows represent an emerging type of such workflows which aim to integrate multiple embedded workflows from potentially different workflow systems to achieve complex experimentation. Workflow interoperability plays a profound role in achieving this objective. This paper is focused at formalizing definitions of different types of workflows and meta-workflows to facilitate improved understanding and interoperability. It also includes thorough formalization of the coarse grained workflow interoperability approach highlighting the role of workflow systems. The paper presents a case study from Heliophysics which successfully demonstrates the use of technologies developed to realize the concepts of meta-workflows and workflow interoperability within a science gateway environment.**

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

I. INTRODUCTION

Scientific workflows represent complex computational experiments conducted by scientists focused at identifying and addressing scientific problems across diverse subject domains. Such experiments usually involve carrying out analysis of large volumes of data and typically run their processes on 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. A typical scientific workflow is composed of one or more individual tasks each of which requires certain input and generates relevant output after performing its intended function.

An interesting and emerging trend in workflow development is composing workflows from one or more sub-workflows. These complex workflows, or *meta-workflows*, may utilize existing workflows from libraries; incorporating such existing workflows as components of the meta-workflow for faster and more efficient development and reusability. Meta-workflows engage complex orchestration of applications which may span across multiple domains. For such complex workflows the nodes represent a combination of workflow jobs and sub-workflows which can host multiple tasks within them. Our focus in this paper is to investigate the challenges encountered in facilitating such complex meta-workflows. In particular, we emphasize on the definition and analysis of different types of meta-workflows and highlight the role of workflow interoperability in different approaches for meta-workflow creation and execution.

The rest of the paper is organized as follows: Section 2 introduces 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 3 briefly describes the fundamental technologies to implement these concepts followed by a description of the scientific use cases to demonstrate our experience with meta-workflows in Section 4. Section 5 describes the related work. Section 6 concludes the paper and lists our future endeavors.

II. 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 1a 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 the workflow are individual jobs that are executed and managed by a single workflow system such as P-GRADE [1], Galaxy [2] or Taverna [3] etc. Fig 1a 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 host workflow system (WS1) as demonstrated by Fig 1b. 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 1c where the two embedded workflows are from workflow systems WS2 and WS3, respectively.

In order to draw a formal representation of the meta-workflow concept, we follow the definitions and guidelines adopted by [4]. 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 and CG1 and CG2 represent the two embedded graphs, then G is a meta-workflow composed of CG1 and CG2 if

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

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

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

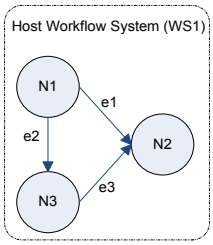

Fig 1a: A single workflow

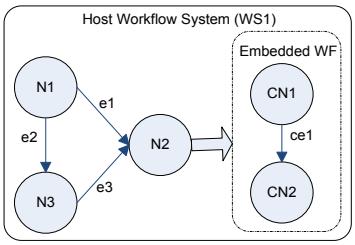

Fig 1b: Native meta-workflow

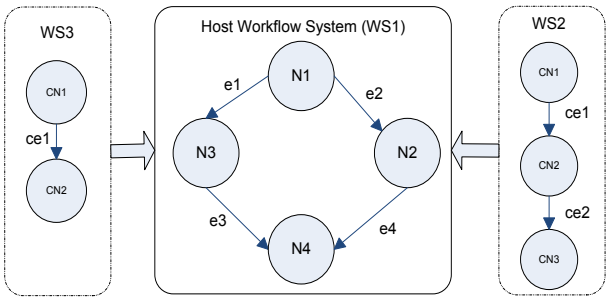

Fig 1c: Non-native meta-workflow

A. Approaches for Non-native Meta-workflow Creation and Execution

In order to create a non-native meta-workflow 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. We explain each of these in more detail below:

i. Coarse-Grained Interoperability Approach: The CGI approach is focused at defining a workflow engine based interoperability concept independent of the workflow system allowing efficient management of workflows from different workflow systems [4]. A fundamental concept within the CGI approach is the native and non-native workflows. The host workflow engine and its workflows are considered native whereas the all other workflows and workflow engines are called non-native. Following this terminology, the non-native workflows and workflow engines are managed as legacy applications by the native workflow system.

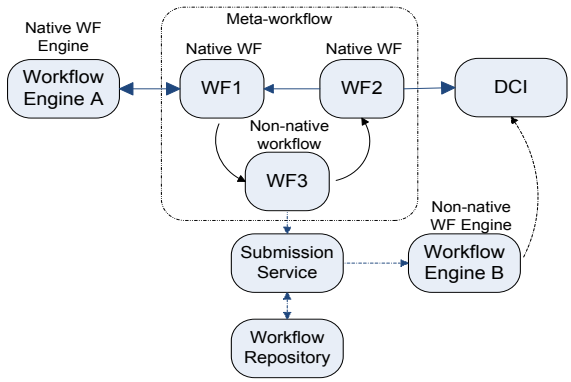

Fig 2: Coarse-grained workflow interoperability usage scenario

The CGI approach requires two important components for successful implementation, i.e. a repository to store/share workflows, and a submission service to understand and manage the formal descriptions of non-native workflows and workflow engines. In a typical usage scenario, the formal description of workflows and workflow engines is published by respective developers in a repository. This formal description is used to wrap the workflows and workflow engines in the form of services. Among other attributes, this description also contains the data required to execute the workflows. When a host workflow engine is encountered with a non-native workflow, it forwards the workflow ID to the submission service. The submission service uses this workflow ID to retrieve the respective formal description from the repository. This formal description includes identification of the respective workflow engine and directions as to execution. Finally, the workflow is submitted to the workflow engine which manages the overall execution of the workflow. Fig 2 presents the sequence of events to execute a non-native workflow using the repository and the submission service as described above.

Borrowing our formal definition of meta-workflows from [4],

WFmeta = $\{J_1 \ldots J_k, WF_{nat1} \ldots WF_{natl}, WF_{nnt1}, .. WF_{nntm}\}$,
where J_{nath} : native job h = 1 . . . k
WF_{nati} : native workflow i = 1 . . . l
WF_{nntj} : non-native workflow j = 1 . . . m.

Although the above definitions present a limited view of the meta-workflow concept, this paper envisages to achieve more comprehensive formalization of the meta-workflow concept and the CGI approach for creation and execution of such workflows. This formalization is envisaged to make significant contribution in improving the usability of the existing meta-workflows by highlighting the requirements of the CGI based workflow interoperability approach. Our effort is inspired by the formal definitions for graphs presented in [5] and is achieved using Z notation [6]. The data models used for our formal definition are:

WF_ID : the unique workflow ID
WF_N : set of native workflows
WF_NN : set of non-native workflows
WF_EN_N : set of native workflow engine(s)
WF_EN_{NN} : set of non-native workflow engine(s)

The formalization for the CGI approach using Z-notation is presented in Fig 3.

CGI based Meta-workflow Execution

$\Delta Meta\_Workflow\_Execution\_CGI$
$WF\_ID? : \mathbb{N}$
$wf_n? : WF\_N$
$wf_{nn}?: WF\_NN$
$wf\_en_n?: WF\_EN_N$
$wf\_en_{nn}?: WF\_EN_{NN}$

$WF\_ID > 0$
$WF\_N \neq <>$
$WF\_NN \neq <>$
$WF\_EN_N \neq <>$
$WF\_EN_{NN} \neq <>$

If $wf_i \in WF\_NN$
$wf_{nn} = wf_i$
$wf\_en_{nn} = wf_i\ (WFeng)$
$Execute\_Submission\_Service(wf_{nn}, wf\_en_{nn})$

Fig 3: Formal description of the CGI approach using Z

ii. 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 its computational tasks' implementation technologies such as the method to call a web service and executing an application on a specific machine.

As part of the FGI approach, the abstract part of the workflow is transformed using an Interoperable Workflow Intermediate Representation (IWIR) as illustrated by Fig 4. 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 [7]. The concrete part of the workflow is not transformed but is bundled with the IWIR representation of abstract part to form an IWIR bundle.

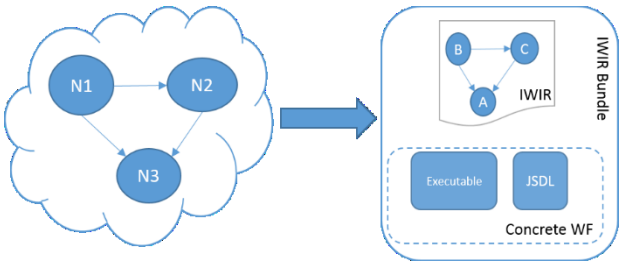

Fig 4: Transformation of workflows for FGI approach

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. 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 make significant contribution in facilitating workflow developers to design new workflows by enabling them to comprehend with the attributes and semantics of each type of meta-workflow.

i. Single job meta-workflow: This type of meta-workflow 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 5 presents a graphical representation of this workflow type.

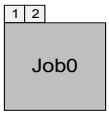

Fig 5: Single job meta-workflow

As we have emphasized as part of our formalization in [4], jobs can be native and non-native therefore we start with the formalization of a native job. Using our definitions of meta-workflows presented in the previous section, a single native job workflow can be represented as:

$J_n = \{J_{abs}, J_{cnf}, J_{cnr}, J_{ennat}\}$ - (Eq. 1)

where J_n : a native job,
J_{abs} : the abstract description of the job,
J_{cnf} : the configuration of the job,
J_{cnr} : the implementation or executable of the job
and J_{ennat} : a native workflow engine for the job

Furthermore, the case where single job native meta-workflow is a native workflow can be described as

$WF_n = \{WF_{abs}, WF_{cnr}, WF_{cnf}, WF_{enn}\}$ - (Eq. 2)

where WF_{enn} represents a native workflow engine and WF_n represents the workflow to be a native workflow.

In order to formalize a non-native workflow, we first describe a non-native job as

$J_{nn} = \{J_{abs}, J_{cnf}, J_{cnr}, J_{ennnat}\}$ - (Eq. 3)

where J_{nn} represents the non-native job and J_{ennnat} represents non-native workflow engine for the job. Further to our formalization for FGI approach in the previous section, $J_{nn} \in$ TASKS where TASKS represents the set of jobs within a workflow. Now, a single job non-native workflow can described as

$WF_{nn} = \{TASKS, WF_{abs}, WF_{cnr}, WF_{cnf}, WF_{ennn}\}$ - (Eq. 4)

where WF_{nn} represents the non-native workflow and WF_{ennn} represents a non-native workflow engine.

ii. 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 6 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 as:

$WF_{lmj} = \{J_1,\ldots,J_m\} = \{J_{n1}\ldots, J_{nk}, J_{nn1} \ldots, J_{nny}, WF_{n1},\ldots, WF_{ny}, WF_{nn1}, \ldots, WF_{nnz}\}$ - (Eq. 5)

where J_j : job of linear multi-job workflow, j = 1.., m
J_{ni} : native job, i = 1 … k
J_{nnp} : non-native job, p = 1… y and refers to the formal description presented in Eq. 3 and 4.
WFnp : native workflow $\in J_j$ p = 1,.., y
WF_{nnt} : non-native workflow, t = 1… z

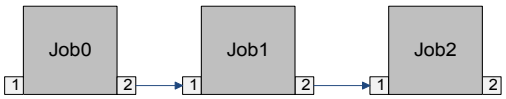

Fig 6: Linear multi-job meta-workflow

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 $e = (o,i)$ where o is the output port and i is the input port. Consequently if P_o is the output port of job J and P_i is the input port of job J.

$$e = (P_o, P_i) \quad \text{- (Eq. 6)}$$

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

$$e_i = (P_{xo}, P_{yi}) \quad \text{- (Eq. 7)}$$

where P_{xo} represents the output port of J_x and P_{yi} represents the input port of J_y.

iii. Parallel multi-job meta-workflow: This is a workflow in the native workflow system that includes parallel branches. One or more of these branches can include one or more non-native workflows. Parallel multi-job meta-workflows are composed of several linear multi-job meta-workflows as presented in Fig 7.

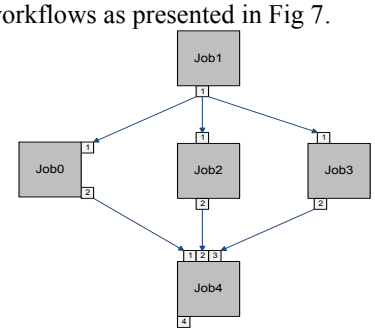

Fig 7: Parallel multi-job meta-workflow

Utilizing formalization from Eq. 5, the parallel multi-job meta-workflow can be described as

$WF_{pmj} = \{J_1,\ldots,J_m\} = \{J_{n1}\ldots, J_{nk}, J_{nn1} \ldots, J_{nny}, WF_{n1}, \ldots, WF_{ny}, WF_{nn1}, \ldots, WF_{nnz}\}$ – (Eq. 8)

where J_j : job of parallel multi-job meta-workflow, j =1,…,m

J_{ni} : native job, i = 1 … k
J_{nnp} : non-native job, p = 1… y
WF_{np} : native workflow $\in J_j$, p = 1,...,y
WF_{nnt} : non-native workflow, t = 1… z

Furthermore, from the formalization for the linear multi-job meta-workflow in the previous section, we define an edge in (x,y).

For a parallel multi-job meta-workflow, the nature of connections among individual jobs is different to that for a linear multi-job workflow defined in the previous section.

Let J_s represent a split job (a job that is followed by several parallel worker jobs), J_w represent a worker job. Then from Eq. 7, a parallel multi-job meta-workflow must contain an edge

$e_g = (P_{go}, P_{wxi})$, where
P_{go} : the output port of the split job
J_{wx} : a worker job and x = 1 … k
P_{wxi} : the input port of the worker job J_{wx}

Furthermore, let J_m represent the merge job (a job that merges the results of parallel workers jobs), then P_{mi} represents an input port for the merge job where i = 1 … k such that k = number of worker jobs. Therefore, a parallel multi-job meta-workflow must also have an edge e_m such that

$e_m = (J_{wxo}, P_{mi})$, where
J_{wxo} : the output of a worker job J_{wx} and x = 1 … k
P_{mi} : an input port of the merge job J_m and i = 1 … k

iv. Parameter sweep meta-workflow: This represents a parameter sweep workflow in the native workflow system that includes one or more non-native workflows. The parameter sweep meta-workflow has a generator job which produces a number of inputs each to be consumed by a worker job. The collector job then 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 are dynamic as compared to the parallel multi-job workflow as presented in Fig 8. 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.

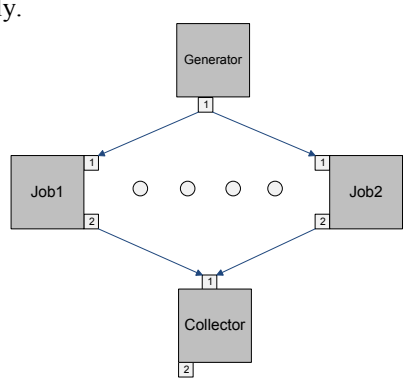

Fig 8: Parameter sweep meta-workflow

Due to the similarity with the parallel multi-job workflow, parameter sweep workflow can be described using Eq. 8 as:

$WF_{ps} = \{J_1,\ldots,J_m\} = \{J_{n1}\ldots, J_{nk}, J_{nn1} \ldots, J_{nny}, WF_{n1}, \ldots, WF_{ny}, WF_{nn1}, \ldots, WF_{nnz}\}$ - (Eq. 9)

where J_j : job of parallel multi-job meta-workflow, j =1,…,m

J_{ni} : native job, i = 1 … k
J_{nnp} : non-native job, p = 1… y
WF_{np} : native workflow $\in J_j$, p = 1,...,y
WF_{nnt} : non-native workflow, t = 1… z

Furthermore, the connections between the generator, collector and the worker jobs for a parameter sweep meta-workflow can be described in similar manner i.e.

$e = (J_{go}, J_{wxi})$, where
J_{go} : the output port of the generator job
J_{wx} : a worker job and x = 1 … k
J_{wxi} : the input port of the worker job *Jwx*

Furthermore, let J_c represent the collector job then J_{ci} represents an input port for the collector job where i = 1 … k such that k = number of worker jobs. Therefore, a parameter sweep meta-workflow must also have an edge *ec* such that

$e_c = (J_{wxo}, J_{ci})$, where

J_{wxo} : the output of a worker job J_{wx} and x = 1 … k

J_{ci} : an input port of collector job J_c and i = 1 … k

III. The SHIWA Simulation Platform

The SHIWA Simulation Platform (SSP) enables data, infrastructure and workflow interoperability at both coarse- and fine-grained level. It supports the whole workflow life-cycle addressing the challenges of (i) executing workflows of different workflow systems as non-native workflows (ii) combining workflows of different workflow systems into meta-workflows and (iii) running these workflows on different DCIs. Currently, the simulation platform provides production-level CGI support for the following workflow systems: ASKALON, Galaxy, GWES, Kepler, MOTEUR, Pegasus, ProActive, Taverna, Triana and WS-PGRADE.

The simulation platform contains a science gateway (SHIWA Science Gateway), a submission service (SHIWA Submission Service), a workflow repository (SHIWA Repository) and a data transfer service (Data Avenue Service). The platform is presented in Fig 10. The SHIWA Science Gateway is built on the gUSE/WS-PGRADE technology. It incorporates the SHIWA Portal, the WS-PGRADE Workflow System and the DCI Bridge service. The portal offers a graphical workflow editor, a workflow engine and workflow execution monitor. The gateway is integrated with the WS-PGRADE Workflow System which is used as native workflow engine in the simulation platform. The DCI Bridge service provides access to different Distributed Computing Architectures such as clouds, clusters, desktop and service grids and supercomputers. The SHIWA Repository stores binaries, configuration and default data and meta-data of workflows and workflow engines. Workflow and workflow engine developers can publish (or export or upload), edit, and delete workflows and workflow engines through the repository GUI. To support execution of non-native workflows the SHIWA Submission Service first, 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 used by different workflow systems.

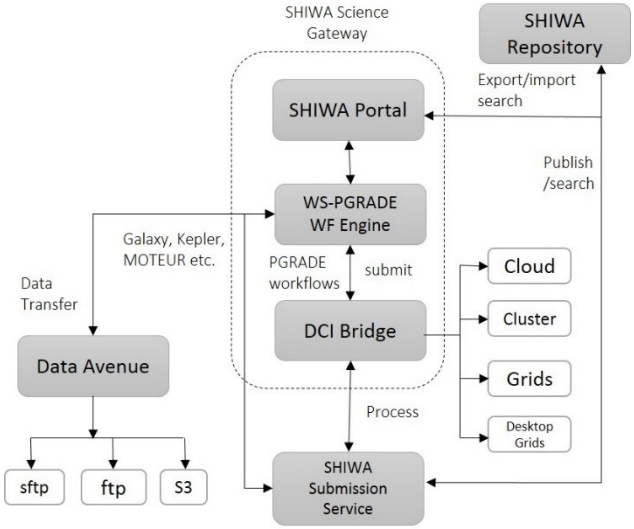

Fig 9: SHIWA simulation platform

IV. The Scientific Use Case

The results of the research presented in this paper have been adopted by diverse scientific disciplines. In this section, we present the experience of the Heliophysics community mapping their use-cases to the formal descriptions described earlier in this paper. The details of this use case have been presented in [8].

Case from Heliophysics – Propagation of Type II CMEs

Coronal Mass Ejection (CME) are extremely relevant phenomena that consist in great masses of charged particles emitted from the Sun and hitting bodies of the Solar System. When Earth is hit by a CME, its environment is disturbed resulting in potentially dangerous effects on satellites, information networks and power distributions. The simulation of propagation of CMEs is quite complex as the speed is influenced by various factors and the physics governing the phenomena are complex.

Furthermore, evaluation of initial conditions of the simulations may be too complicated to assess precisely. This investigation is focused on fast CMEs and it uses a simple ballistic model with corrections. It is assumed that fast CME will slow down due to Solar Wind drag during the propagation. Based on this assumption, the propagation model is corrected with values of the speed of Solar Wind at target (the Earth).

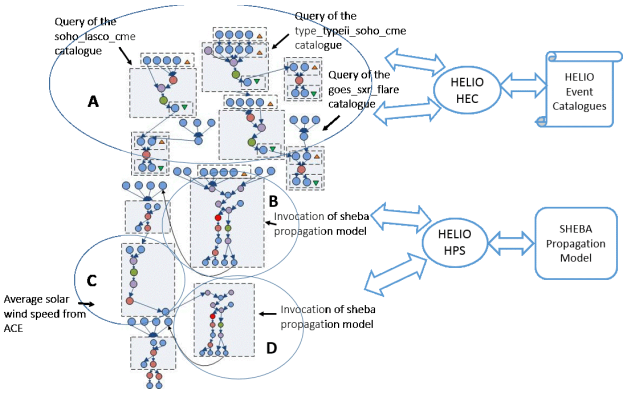

Fig 10: Meta-workflow for the propagation of type II CME

Linear multi-job meta-workflow: Investigation of the fastest CME in a given period of time is executed with the linear multi-job meta-workflow described in Fig 10. The embedded Taverna workflow is composed of four main phases:

A. Evaluation of initial parameters for the propagation. Values obtained from the HELIO Event Catalogue (HEC) that, in turn, accesses table with the details of various solar events. The fastest CME within the given time range is selected and the initial parameters obtained from the event catalogues.
B. Execution of the SHEBA Propagation Model on the HELIO Processing Service (HPS).
C. Comparison of the results of the propagation model at target (Earth) with the speed of the solar wind and corrections of the propagation of the initial parameters.
D. Execution of the SHEBA Propagation Model on the HPS with the corrected parameters.

Parameter sweep meta-workflow: This meta-workflow finds and propagates the fastest CME over a given time range. The WS-PGRADE parameter sweep extension (Fig 11) of the Taverna meta-workflow performs the same operation over many time ranges. A simple WS-PGRADE workflow creates a series of time ranges from a general time range and granularity and invokes the Taverna meta-workflow for each of them. As each execution of the Taverna meta-workflow is independent from each other, they can be executed in parallel.

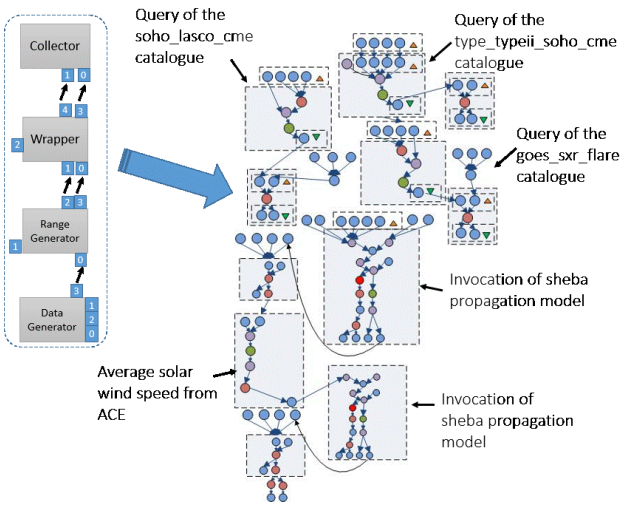

Fig11: Parameter sweep meta-workflow in WS-PGRADE

V. RELATED WORKS

The usage of scientific workflow paradigm has been widely adopted to address problems across widespread domains such as Computational Chemistry [9] [10], [11], Astrophysics [12], or Bioinformatics [13]. 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, MOTEUR[14], Galaxy, and P-GRADE.

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* [4].

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) [15] 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 [4]. The FGI concept is built on workflow language translation using the Interoperable Workflow Intermediate Representation (IWIR) for common workflow description [7]. 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.

The ClowdFlows [16] 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. [13] 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 [17] 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.

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.

VI. CONCLUSIONS

The paper has contributed towards the definition and analysis of meta-workflow approaches for complex scientific meta-workflows. It has focused on the formal definition of different types of CGI-based meta-workflows including formalizing the CGI-based meta-workflow execution. The paper has also presented experimentation for heliophysics as a proof of concept for the approach presented in the paper. We aim to address challenges with respect to FGI based meta-workflow approach as part of future work.

VII. ACKNOWLEDGEMENTS

We acknowledge the contribution of Dr. Gabriel Pierantoni (TC-HPC), Dr. Eoin Carley (TCD-Physics) and Dr. Byrne (Rutherford Appleton Laboratory) for this research.

REFERENCES

[1] A. Kertész, G. Sipos and P. Kacsuk; Brokering multi-grid workflows in the P-GRADE portal, in: Euro-Par 2006: Parallel Processing, vol. 4375, Springer, Berlin, 138–149, 2007.

[2] Giardine, B., Riemer et al.; Galaxy: A platform for interactive large-scale genome analysis. Genome Research, 15(10):1451-5, 2005.

[3] Oinn, T., Addis et al.; Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics, 20(17):3045-54, 2004

[4] Terstyanszky, G. et al; Enabling Scientific Workflow Sharing Through Coarse Grained Interoperability in the Future Generation Com[puter Systems Vol 37, 46-59, 2014.

[5] Toda, Y.; The Formalization of Simple Graphs, Formalized Mathematics, vol.5, no. 1, 137-144, 1996

[6] ISO/IEC 13568, "Information Technology- Z formal Specification Notation- Syntax,TypeSystemandSemantics:InternationalStandard", 2002

[7] Plankensteiner K. et al; Fine-Grained Interoperability of Scientific Workflows in Distributed Computing Infrastructures, in the Journal of Grid Computing, Vol 11(3), 429-455, 2013

[8] 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

[9] Herres-Pawlis, S. et al.; Quantum chemical meta-workflows in MosGrid in the Concurrency and Computation: Practice and Experience 27(2), 344-357, 2015.

[10] Herres-Pawlis, S. et al.; 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

[11] Herres-Pawlis, S. et al.; User-friendly metaworkflows in Quantum Chemistry, in the IEEE International Conference on Cluster Computing, 1-3, 2013.

[12] Becciani, U. et al.; Science Gateway technologies for the astrophysics community, in the Concurrency and Computation: Practice and Experience, 2014

[13] 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.

[14] 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

[15] SHIWA: SHaring Interoperable Workflows for large-scale scientific simulation on Available DCIs. http://www.shiwa-workflow.eu, 2011.

[16] 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

[17] Gil, Y., Ratnakar, V., Dellman, E.; Wings for Pegasus: A Semantic Approach to Creating Very Large Scientific Workflows, in OWLED 2006.