Automating Multivariable Workflow Composition for Model-to-Model Integration

Raul Alejandro Vargas-Acosta
Computer Science
The University of Texas at El Paso
El Paso, Texas, USA
ravargasaco@miners.utep.edu

Luis Garnica Chavira
CyberShare Center of Excellence
The University of Texas at El Paso
El Paso, Texas, USA
lagarnicachavira@utep.edu

Natalia Villanueva-Rosales
Computer Science
The University of Texas at El Paso
El Paso, Texas, USA
nvillanuevarosales@utep.edu

Deana D. Pennington
Earth, Environment and Resource Sciences
The University of Texas at El Paso
El Paso, Texas, USA
ddpennington@utep.edu

***Abstract*— Many societal-relevant challenges, including environmental ones, require comprehensive approaches that integrate decoupled data, models, and perspectives. Integrating data and models is critical for these approaches but can also become cumbersome. Computational workflows are widely used to integrate heterogeneous data and computational processes within or across domains. However, creating computational workflows may require computational and domain expertise not necessarily possessed by potential users. This paper presents our efforts to enable automated multivariable workflow composition implemented as a workflow composer in the Sustainable Water for Integrated Modeling (SWIM) platform. We describe the uninformed search algorithm used in the workflow composer and an initial evaluation with a case study that requires integrating two water (balance) models that cover the Middle Rio Grande in the U.S. Southwest region. Preliminary results show that the evaluation of integrating models should not only consider the technical and scientific perspective but also how users understand and use the results of these complex systems. Efforts toward automating the model-to-model integration can significantly support scientific endeavors and decision-making by enabling various stakeholders to use scientific models.**

Keywords—workflow composition, model-to-model integration, cyberinfrastructure, democratizing computational workflows, containers, microservices

I. Introduction

Many of the most challenging real world-problems confronting society, such as environmental issues and sustainable development, are tightly coupled and achievable only with unprecedented integration of perspectives across disciplinary, professional, cultural, institutional, and political boundaries, as well as communication of findings to decision-makers [1], [2], [3]. Integrating models to address such complex problems is frequently done ad hoc [4]. Models have data inputs that can be complex, and they generate voluminous data outputs that must be analyzed and/or visualized to be understood. Moreover, a single model might not produce the desired information and the execution of several models is needed, which requires keeping track of which inputs resulted in which outputs. Software solutions can ease the burden of data and model integration, allowing scientists to rapidly construct complex analyses that better represent real-world problems and enable findings to be more easily communicated to decision-makers.

Computational workflows enable the integration of data and models across different domains [5]. At the core level, computational scientific workflows describe tasks and data needed to address a particular problem [6]. Computational workflows are widely used across domains that require expensive computational processes.

Creating a workflow usually starts at a conceptual level with the use of abstract representations such as Data Flow Diagrams (DFD) and Directed Acyclic Graphs (DAGs). This abstract representation can be mapped to a workflow structure that allows users to manage relevant processes through workflow-management systems (WMS) [7]. Declarative workflows can then be serialized to a target WMS system using programmatic libraries or following tool-specific syntax and structure. The workflow specification may also require metadata to locate data and jobs across distributed computational environments, along with data transfer protocols and credentials. The disparity of workflow specification languages across WMSs was addressed by standardization efforts such as the Common Workflow Language (CWL) [8].

Despite broad access and standardization efforts, users may require domain and technical expertise for manually building workflows describing complex processes. Opportunities to provide additional tools supporting the creation of workflows and automate tasks have been previously identified [7].

Cloud services currently provide users easy access to computational resources where the management of computing nodes, networking, storage and power resources rests solely on cloud-service providers [9]. Cloud services also provide Web-based interfaces to simplify the deployment and use of custom applications. Burkat et al. identified the model of serverless computing (i.e., applications running on the cloud) as a viable execution model for scientific workflows [10].

This material is based upon work supported by the National Science Foundation (NSF) under Grant No. 1835897. This work used resources from the Cyber-ShARE Center of Excellence, which is supported by NSF Grant No. HRD-1242122.

This paper presents an approach using search strategies to build workflow plans on-the-fly supported by a cloud-ready architecture that has been applied for model-to-model integration in the Sustainable Water Through Integrated Modeling (SWIM) Platform[1]. In particular, we describe our efforts to: i) automate model-to-model workflow creation, ii) serialize the workflow to CWL to use existing WMS, and iii) attenuate barriers to entry for users by simplifying the payload structure of workflow requests and workflow outputs. This work can contribute to the democratization process of scientific modeling, in alignment with the FAIR principles [11], by making scientific workflows more Findable, Accessible, Interoperable, and Reusable by experts and non-expert stakeholders.

This paper is organized as follows: first, we provide background on the transition of scientific computational workflows to cloud-based infrastructure. Second, we explore related work that uses rich metadata and semantics for workflow composition, complementing our efforts. Third, we present our approach to automating workflow composition, including a description of the core algorithm and an abstract example. Next, we present an end-to-end implementation of a model orchestration microservice pool illustrated with a water-domain scenario with preliminary evaluation findings. A discussion with lessons learned follows. Last, we provide conclusions and future work.

II. Background

Scientific domains such as physics, astronomy, and biomedicine require complex workflow composition and execution management implementations [5], [12]–[17]. In the biomedical research field, Kotliar et al. identified over 100 workflow management systems, each with its way of specifying workflow pipelines [18]. These different specifications make workflow reproducibility across WMSs, particularly domain-specific ones, a challenging task.

Generic WMSs provide a higher level of abstraction for creating and executing workflows regardless of the application domain. In addition, many of these systems are gradually migrating to workflow specification standards such as the CWL[2]. This transition can provide a seamless workflow specification compatible with multiple WMSs, thus facilitating the reproducibility and interoperability of computational workflows across application domains.

The underlying architecture of traditional WMSs uses HPC task management systems such as SLURM [19] and HT Condor [20]. WMSs built on top of these systems were initially designed for scientists to deploy workflow pipelines in HPC environments such as local campus clusters and computing centers. Workflow processes typically rely on (worker) nodes that execute tasks provided by the WMS queue [10]. With the emergence of cloud computing, researchers and WMS providers acknowledge the viability of migrating scientific workflows to the cloud with challenges such as cost and performance requiring further research [21]–[23].

The potential for computational cloud services for scientific workflows has been acknowledged for over 10 years. The first cloud platform rollouts were focused on web and business applications. Back then, available services did not focus on supporting large-scale workflow applications [24]. Today, Cloud platforms like Microsoft Azure[3], Amazon Web Services (AWS)[4], and Google Cloud Platform (GCP)[5] facilitate the deployment of custom software applications in a few steps, eliminating the time-consuming task of setting up the environment.

The emergence of containerization technology such as Docker[6], Mesos Containerizer[7], LXC Linux Container[8], and Podman[9] enabled the portability of applications across different cloud or on-site environments. Containers are considered a possible solution for some workflows that incorporate heterogeneous systems [6]. Cloud providers now offer different solutions to deploy, host, and manage containers on their platforms (e.g., Amazon EKS, Amazon ECS, Azure Kubernetes Service, Google Kubernetes Engine, and Google Cloud Run). In addition, containerization has fostered the creation of microservice architectures, replacing traditional monolithic backends. Microservices are self-contained and independent services that focus on specialized tasks [25] that can be reused and orchestrated across single or cross-platform applications. I. Salvadori et al. recognize the advantages for adopting microservices, including deployment and maintenanceas well as the complexity created that requires better communication and cooperation [26].

Cloud computing is further driving the creation and extension of WMSs to support workflow management in the cloud and distributed environments. The REANA platform was created in 2019 with data analysis and reproducibility in mind. It was developed with the Kubernetes-orchestrated execution backend to run containerized pipelines on the cloud [27]. Recent updates of REANA incorporate a hybrid approach that includes access to the traditional HT Condor backend [28]. Vahi et al. describe their progress in supporting container technologies in the Pegasus WMS. Their approach allows the automatic deployment of containers and job data at workflow runtime on platform-agnostic computing environments [13]. Pegasus currently supports container technology from Docker, Singularity [29], and Shifter [30].

III. Related Work

In our previous work as part of the Earth, Life, and Semantic Web Project (ELSEWeb), we addressed data-to-model integration for species distribution models (SDMs) [31]. Our approach proposed a semantic-based cyberinfrastructure to automate the retrieval and manipulation of data to be consumed

[1] https://purl.org/swim
[2] https://www.commonwl.org/
[3] https://azure.microsoft.com/
[4] https://aws.amazon.com/
[5] https://cloud.google.com/
[6] https://www.docker.com/
[7] https://mesos.apache.org/
[8] https://linuxcontainers.org/
[9] https://podman.io/

by SDMs. ELSEWeb integrated data from the Earth Data Analysis Center (EDAC)[10] with species-distribution modeling services provided by Lifemapper[11]. The data and modeling services were semantically described by extending upper-level ontologies such as DCAT[12], OBOE[13], SIO[14], and PROV-O[15]. A semantic bridge (i.e., a pipeline) was built to connect data and model provider services with a pool of semantically-enhanced, data-transformation services. The pipeline description is consumed as a SPARQL query by the Semantic Automated Data Integration framework (SADI) [32], which enables the automated reasoning to streamline the generation and execution of SDMs [33].

In 2011, Gil et al. presented Wings, a workflow creation system that uses semantic representation and planning techniques to create workflow templates and instances. The process requires the active participation of a domain expert to define a detailed description of input variables and an abstract workflow. The abstract workflow includes semantic descriptions that allow Wings to decipher, locate, or serialize the workflow into an executable instance. A WMS can then execute the resulting workflow specification [34].

Subsequently, the MINT project was proposed by Gil et al. to assist users with cross-disciplinary model integration building on existing tools, including CSDMS [35], BMI [36], GSN [37], WINGS [34], Pegasus [23], Karma [38], and GOPHER [39]. MINT's approach uses semantic representations to describe model requirements and data characteristics, automatic planning through abductive reasoning techniques, a data discovery and integration framework, and machine learning algorithms for model parameterization [40]. Their current implementation[16] is being used to explore the role of weather on food availability in selected regions of the world [41].

Kasalica et al. present the Automated Pipeline Explorer (APE), a framework for automated workflow composition. Domain knowledge in APE is represented through annotations and taxonomies. The required input/output data and other restrictions are represented with temporal constraints using Semantic Linear Time Logic (SLTL). SLTL expressions are translated into propositional logic formulas that a SAT solver can solve, and the results are used to construct the workflow [42].

Klampanos et al. provide an overview of DARE [43], a framework that provides a set of tools for the development, discovery, and sharing of workflows. DARE contains a Python API for describing workflows, the s-ProvFlow framework for capturing the execution flow of jobs within a workflow, a registry for storing workflow entities in both dispel4py and CWL, an execution API for workflows, and tools for testing and debugging workflows. DARE can execute workflows described using dispel4py [44] or CWL.

The previous approaches rely on a domain expert to provide a semantic description of the data, the domain, and the tools used for data processing (e.g., transformations and models). In addition, users are required to describe an abstract representation of the pipeline. ELSEWeb uses SPARQL to represent pipeline requirements, Wings provides a Graphical User Interface to build a semantic representation, and APE uses natural-language templates aligned to logical propositions. In the MINT project, Wings can generate workflow plans for the Pegasus WMS. APE currently generates workflows as shell scripts, and support for the CWL is planned as part of their future work. DARE makes use of dispel4py to support CWL. In contrast, ELSEWeb uses SADI to orchestrate semantic web services.

Our approach follows principles of simplicity, abstraction, and decoupling across the SWIM platform. The storage of processing services metadata and data elements is simplified through a semi-structured, unified JSON schema introduced in [45]. The JSON format is used across SWIM for storage, data transfer, and generation of web-based visualizations. JSON can be further extended as JSON-LD to align with semantic annotations without impacting the current system functionality.

We extended the SWIM microservice architecture to automate workflow generation for model-to-model integration. The following section describes the multivariable workflow composition algorithm with an abstract use case and the end-to-end implementation. All microservices are deployed as docker containers as part of the SWIM ecosystem of webservices.

IV. Automated Workflow Composition

SWIM's current approach to the automated composition of workflows uses a breadth-first search strategy. A directed acyclic graph is built by expanding sibling nodes first; as opposed to depth-first search where child nodes are expanded first [46]. In Fig. 1, a computational process (p) is represented as a node that needs to be expanded. Variables produced by this computational process enable the exploration of additional computational processes that can consume the generated data as part of their inputs. In our scenario, computational processes can produce and consume multiple data elements (e), which we also refer to as variables in the rest of the manuscript and thus the name of multivariable workflow composition.

Algorithm 1 shows the multivariable workflow composition in SWIM as pseudocode. This algorithm can automatically compose a workflow of processes that consume and produce multiple data elements. The algorithm composes a scientific workflow by iteratively identifying candidate processes that can be executed based on available data elements, excluding processes that have been analyzed (line 10). For each candidate process, the data variables generated from such process are added to a set of collected data variables with information about the iteration (i.e., step) and the process that produced it (lines 14 – 21). The algorithm continues to iterate until all desired data elements (i.e., target variables) are collected, or there are no processes available to explore (lines 11 – 13). If multiple paths can lead to the generation of the target variables, only the first

[10] https://edac.unm.edu
[11] https://lifemapper.ku.edu
[12] http://www.w3.org/ns/dcat
[13] https://bioportal.bioontology.org/ontologies/OBOE
[14] https://bioportal.bioontology.org/ontologies/SIO
[15] https://www.w3.org/TR/prov-o/
[16] https://mint.isi.edu

path found is returned. Fig. 1.A shows a DFD that represents a multivariable workflow example. If a process does not contribute to the final solution (e.g., the process identified as "p8" in Fig. 1.A), that process it is not included in the workflow (lines 26 – 40). Processes that cannot be executed due to a lack of data elements are not included in the analysis (e.g., the process identified as "p9" in Fig. 1.B).

In this example (Fig. 1.A), the target data elements are labeled as "e11" and "e12", user-provided data elements are "e1", "e2", "e3", "e4", and "e5". The target data elements and input data elements form part of the abstract workflow request received as an input. The abstract workflow request also includes metadata for available processes, referred to this example as "ProcessCatalog" that contains information for the processes (i.e., "p1,…,p9"), including their data inputs and outputs. The abstract workflow request is further described in Subsection V.B.

All processes are described in the process catalog. The process catalog includes the data elements consumed by a process and the data elements produced. The process catalog used in our example contains nine processes depicted in Fig. 1; the blue path shows a candidate workflow path for generating the target variables.

The multivariable workflow composition algorithm generated the blue path by first analyzing all desired and available initial data variables (lines 1-2 in Algorithm 1) for the creation of a target and collected data sets, a record of the origins of the initial data variables is created in lines 5 – 8. Based on collected variables and visited process, process nodes that can be executed are collected in line 10, resulting in "p1" being selected. Data variables generated by this process are added to the collected data set, the record of data origins is updated, and "p1" is added as a visited process. This iteration is repeated, and now line 10 expands processes p2, p3, p4, and p5; data variables generated by these processes are added to the collected data set, the record of data origin is updated, and the visited process is updated. In our example, both processes "p3" and "p4" generate the same variables; however, only the first process selected for analysis that generated the variable is used as the origin of the variable. In the third iteration, the processes p6, p7, and p8 are retrieved from the process catalog in line 10, after analyzing their outputs all target variables are generated. In the fourth iteration, line 9 validates that the collected data set contains the target variable and stops iterating. TABLE I shows the possible paths for this abstract example; however, only one path is selected in the algorithm (TABLE I, row 1). Each "step" can include a set of processes that can be executed in parallel.

At this stage, the process graph contains multiple paths for generating target variables and a process that doesn't contribute to the final output. Lines 26 – 40 then use the record of data origin to retrieve the iteration and the process that generated it. The cycle continues for every data collected, given the inputs required by the processes.

Algorithm 1: Multivariable Workflow Composition	
1:	*targetVariable = desired variable provided in the request*
2:	*collectedVarSet = available variable provided in the request*
3:	*iterationNumber = 1*
4:	*visitedProcess* = new Set(String)
5:	**for** each *variable* ∈ *collectedVarSet* **do**
6:	*variable*.addOrigin("request")
7:	*variable*.addStep(0)
8:	**end for**
9:	**while** !*collectedVarSet*.contains(*targetVariable*) **do**
10:	*executableProcesses* = ProcessCatalog.getProcessBy(*collectedVarSet, visitedProcess*)
11:	**if** *executableProcesses*.isEmpty() && !*collectedVarSet*.contains(*targetVariable*) **do**
12:	**return** Exception("Target variable cannot be generated with available processes and initial variables")
13:	**end if**
14:	**for** each *process* ∈ *executableProcesses* **do**
15:	**for** each *variable* ∈ *process*.getOutputs() **do**
16:	*variable*. addOrigin(*process*)
17:	*variable*. addStep(*iterationNumber*)
18:	*collectedVarSet*.add(*variable*)
19:	**end for**
20:	*visitedProcess*.add(*process*.getId*()*)
21:	**end for**
22:	*iterationNumber*++
23:	**end while**
24:	*workflowMap* = new Map(Integer, Process set)
25:	*varToBeCollectedSet = desired variable provided in the request*
26:	**for** each *variable* ∈ *varToBeCollectedSet* **do**
27:	*varMetadata* = *collectedVarSet*.get(*variable*)
28:	**if** *workflowMap*.contains(*varMetadata*.getStep()) **do**
29:	*processSet* = *workflowMap*.get(*varMetadata*.getStep())
30:	**end if**
31:	**else do**
32:	*processSet* = new Set(Process set)
33:	**end else**
34:	*process* = *varMetadata*.getOrigin()
35:	**for** each *variableInput* ∈ *process*.getInputs() **do**
36:	*varToBeCollectedSet*.add(*variableInput*)
37:	**end for**
38:	*processSet*.add(*process*)
39:	*workflowMap*.put(*varMetadata*.getStep(), *processSet*)
40:	**end for**
41:	**return** *workflowMap*

After this process is finalized, the algorithm creates a workflow composed of the process {"p1"} as the first step; {"p2","p3","p5"} as the second step, these processes can be executed simultaneously as there is no data dependency among each other; and {"p6"} as the third and final step.

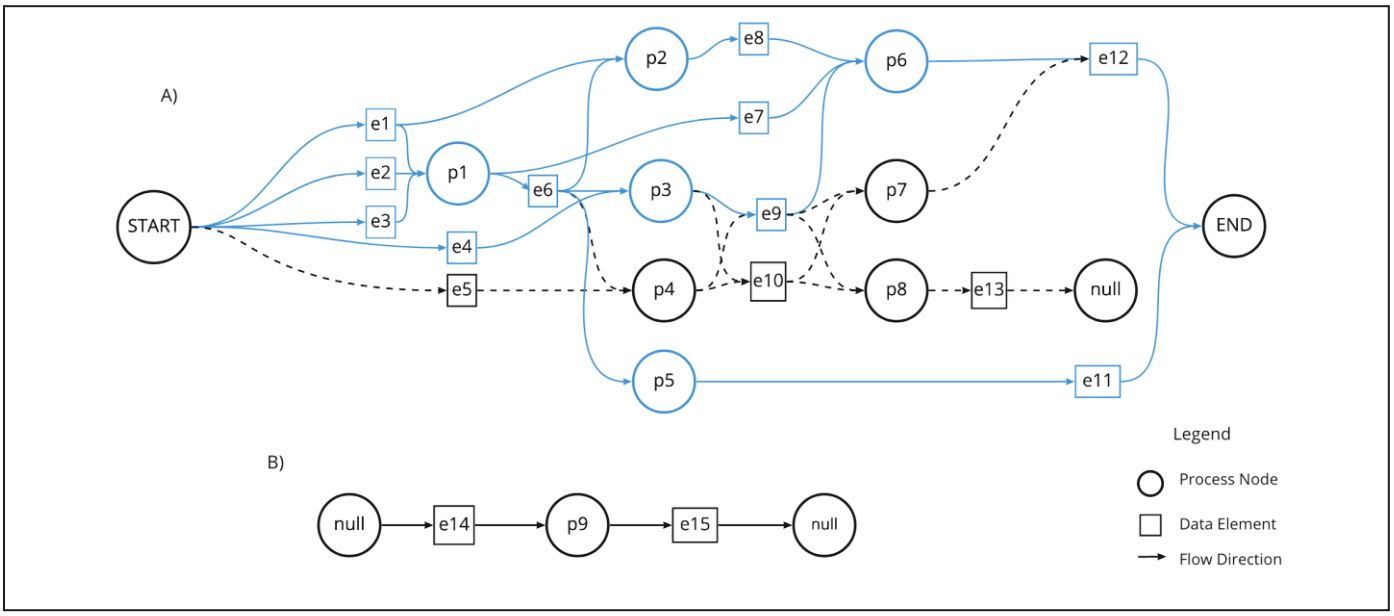

Fig. 1. DFD of a multivariable workflow example. A) Depicts processes analyzed using Algorithm 1. The dashed path passes through candidate nodes, the blue solid path is selected for the workflow. B) Illustrates how processes in the catalog that cannot be executed because input variables are not provided are not visited by the Algorithm 1.

TABLE I. POSSIBLE WORKFLOW PATHS

Path	Step 1	Step 2	Step 3
1	p1	p2, p3, p5	p6
2	p1	p2, p4, p5	p6
3	p1	p3, p5	p7
4	p1	p4, p5	p7

V. END-TO-END MODEL-TO-MODEL INTEGRATION

This section describes our end-to-end solution for a case study that requires model-to-model integration. SWIM's architecture builds upon existing capabilities of WMS, automated workflow planning, and cloud-based infrastructure. Scientific computation processes are encapsulated as webservices and containerized for easy deployment to the cloud or on-site infrastructure. The capabilities of workflow composition and management are also offered as a web service through a container.

A sequence diagram of the SWIM Orchestration Microservice Pool is shown in Fig. 2. A user's workflow request (SWIM workflow request) initializes the sequence. A user's request can be performed programmatically via an HTTP request or through SWIM's Broker Open API documentation. A service orchestration endpoint receives the workflow request.

A. The SWIM Broker Service

The *SWIM Broker Service* [17] is an application-specific service that directs the workflow processing logic into a sequence of internal method calls and external microservices. The *SWIM Broker* first validates user credentials in the form of a JSON Web Token (JWT). The authentication token must be included as part of the workflow request headers. The SWIM workflow request contains a JSON payload that specifies data inputs, outputs of interest, and pre-processing operation rules.

The pre-processing unit of the *SWIM Broker* currently supports two rules: equivalence and default data exclusion. These rules are further explained in section VI.B.

The pre-processing unit performs the following operations: i) assigns a unique identifier to the workflow, ii) applies user-defined rules, iii) generates an abstract model catalog, iv) generates an abstract workflow request, and v) stores mappings between SWIM identifiers and those of the abstract products. The by-products of the pre-processing unit are serialized as

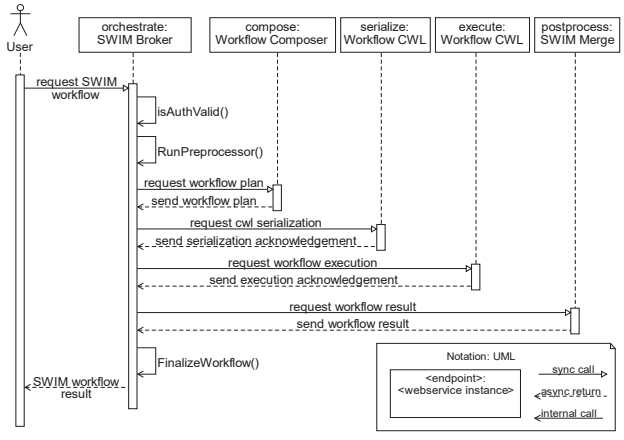

Fig. 2. Sequence Diagram of the SWIM Orchestration Microservice Pool

[17] https://services.cybershare.utep.edu/swim-broker/swagger

JSON payloads and sent over to the workflow composer to request a workflow plan.

B. The Workflow Composer Service

The *Workflow Composer Service* receives the abstract model catalog and the abstract workflow request as input. These two artifacts include platform-independent metadata that identifies models and data elements with general unique identifiers. The abstract model catalog contains the metadata of available modeling and transformation services. The abstract workflow request contains the user-defined payload with the identifiers generated by the pre-processing unit.

The multivariable workflow composition algorithm, described in section IV, is implemented in this service. The computational processes for the case study in section VI are classified as scientific models or data transformation jobs. All the models and transformation jobs are wrapped with a webservice interface in our implementation. Transformation services enable the serialization of data values in the format required by the following scientific model to be executed. Data changes might include units (e.g., Metric to English) or data structures (e.g., different schemas).

The serialization of the workflow plan is generated in JSON format. The serialized workflow plan contains metadata for the execution of every model and its prerequisites (i.e., models or transformation services that need to be executed beforehand). The workflow plan is sent as a response to the *SWIM Broker Service*, which in turn, sends the workflow plan to the *Workflow CWL service*.

The Workflow Composition implementation as a microservice is available online on GitHub[18] and as a docker image on the DockerHub registry[19].

C. The Workflow CWL Service

SWIM leverages the third-party CWL Python API for creating CWL workflows [47]. A CWL serialization can be used to execute workflows in a WMS that uses this same standard (e.g., Pegasus). The CWL API is used to transform a workflow plan serialized as JSON into a CWL workflow and use the workflow management capabilities of the CWL tool. All the processing nodes in SWIM communicate through webservice interfaces. Thus, the *curl* command in the Linux container of the *Workflow CWL Service* is used to send and receive messages using the HTTP protocol.

The *Workflow CWL* implementation as a microservice is available online on GitHub [20] and as a docker image on DockerHub[21].

D. The SWIM Merge Service

The *SWIM Merge Service* is a post-processing microservice that prepares the response payload of the workflow. The *merge service* receives the workflow identifier. From this identifier the service can query the workflow database to retrieve the mappings between SWIM and the abstract workflow. Once the identifiers are translated to SWIM terms, the service retrieves modeling outputs requested by the user. The workflow database also stores provenance of where each output was generated from; this is included as another block in the final payload.

E. Workflow Result Availability

Scenario results generated from each model in the workflow are made available in the Public Scenarios listing[22] with all the features available in the current SWIM web-based interface (e.g., visualizations and sorting of outputs according to role). Workflow scenarios can be identified by the name containing "Custom workflow scenario" followed by a unique identifier. The scenario results can be consulted within the interface and can be modified and executed as a new scenario entry. The web interface is currently limited to executing only one model and scenario at a time. The model orchestration capabilities, described in this manuscript, are currently available only through the *SWIM Broker Service*.

VI. Case Study

A. Overview

This case study aims to support scientists and policy-makers in exploring different scenarios and the effects of short-term management strategies projected into the future. In particular, answering the question: *How does regional reservoir storage behave in an economically optimal water use scenario?*

This case study requires the integration of two heterogeneous models available in SWIM, namely the Water Balance Model (WBM) [48] and the Hydroeconomic Model (HEM) [49]. The coverage area for both models is bounded to the Middle Rio Grande in the Paso del Norte region, which includes the south of New Mexico (NM), West Texas in the US, and the north of Chihuahua in Mexico. The WBM is a regional water supply simulation model driven by: upstream inputs to Elephant Butte Reservoir in NM, local climate, regional water demand, and existing reservoir operation rules. The HEM is an economic optimization model that maximizes profits from regional water use. Both models can take multiple inputs and generate output values for multiple variables.

The metadata corresponding to the models integrated in SWIM, along with metadata about their inputs and outputs are stored on a database instance accessible through the SWIM API[23]. The SWIM Broker pre-processes these catalogs into an abstract catalog representation that is included as payload to the workflow composer.

Fig. 3 shows a DAG diagram of the model-to-model integration scenario derived from our case study (i.e., the composed workflow). Nodes represent a webservice exposing a scientific model (e.g., HE Model) or a transformation job (e.g., swim-assembler). The rectangles represent data elements. The top and bottom boxes correspond to workflow inputs and outputs, respectively. The swim-assembler handles the retrieval of default data values for variables from the modeling database and constructs a model scenario input for model consumption.

[18] https://purl.org/swim/repos/composer
[19] https://purl.org/swim/docker/composer
[20] https://purl.org/swim/repos/cwl
[21] https://purl.org/swim/docker/cwl
[22] https://swim.cybershare.utep.edu/en/public
[23] https://services.cybershare.utep.edu/swim-api/api-docs/

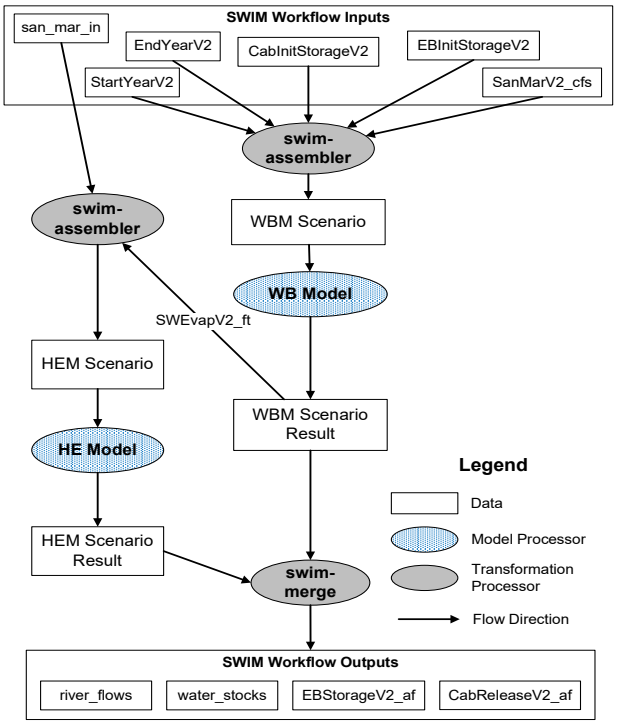

Fig. 3. DAG of a SWIM model-to-model integration scenario derived from the case study where the models HE and WB are integrated.

Below is a description of the inputs used to generate this workflow and the outputs generated, which are also depicted in Fig. 3.

B. SWIM Workflow Input

In this section, we describe the content of the SWIM workflow request payload used in the case study. The payload is divided into three blocks: inputs, outputs, and rules.

Listing 1 shows an excerpt of the SWIM workflow request payload. The first block of the input payload includes data inputs with custom values specified by the user. For each input entry, the "paramName" field carries a unique identifier for the data element. The "paramValue" field is a user-defined value for the parameter. The scenario provided indicates a numeric value of 1994 to the input parameter with the identifier "StartYearV2". The remaining inputs for this scenario are depicted in Fig. 3. The workflow composer implementation supports numeric, table, and time-series data serialized in JSON. As the "paramValue" field is not bounded to a specific data type, this field can potentially reference more complex data input types (e.g., Tiff, Geo-JSON, NetCDF). The outputs block specifies the target output data that should be obtained after the workflow execution. The "varName" field holds a unique identifier for each output. Both input and output blocks can be extended with additional metadata that could enable data transformations such as unit conversions, resolutions, or time-series timesteps.

The rules block defines user-selected rule objects. Currently, rule objects are manually specified by users as a JSON payload through a simple editor available in the SWIM broker. Each rule object describes one rule applicable to a set of variables (i.e., model inputs and outputs). Below is the current notation of a rule object:

```
{ "inputs" : [
            {"paramName" : "StartYearV2",
             "paramValue" : 1994 }],
  "outputs" : [
            {"varName" : "EBStorageV2_af " },
            { "varName" : "water_stocks" } ],
   "rules" : [
            { "excludeDefault" : ["evap_rat_p"] },
            { "equivalence" : ["evap_rat_p", "SWEvapV2_ft"]}]
}
```

Listing 1. SWIM workflow input excerpt for the case study.

{"ruleName": ["variable-1", "variable-2", …]}

Our current implementation supports two rules, the *equivalence* and the *excludeDefault* rules. The *equivalence* rule defines a set of variables to be semantically equivalent, which entails that all corresponding variables satisfy alignment restrictions w.r.t. data format, data structure, measurement units, and resolution. In the case of a time series, further consistency checks should include time range and time-step resolution. Automated data consistency checks are part of ongoing development that can be potentially leverage related work [34]. In addition, automatic generation of data transformation sub-workflows can further facilitate the alignment of variables to satisfy the equivalence rule and other potential rules.

```
{ "@context": "http://purl.org/swim/vocab",
  "metadata": {
      "id":  "2c338b57-88e9-4ea0-a100-227f980a3465",
      "status": "success",
      "type": "Workflow Result" },
   "provenance": [ {
      "entity": "Model Output",
      "generatedAtTime": "2022.05.25.14.06.25",
      "id": "EBStorageV2_af",
      "wasGeneratedBy": "1fb918b3-bf35-40f3-9821-b4d907cd610f" }],
   "resource": [
            {
              "modelID": "7b7ac93638f711ec8d3d0242",
              "varName": " EBStorageV2_af",
              "varValue": "...",
              "varinfo": [
                 {
                  "lang": "en-us",
                  "varCategory": " Storage",
                  "varDescription": "Elephant Butte reservoir storage...",
                  "varLabel": "Elephant Butte Reservoir Storage",
                  "varUnit": "Acre-Feet"
                 },
                 {
                  "lang": "es-mx",
                  "varCategory": "Almacenamiento",
                  "varDescription": "Promedio anual en el volumen...",
                  "varLabel": " Almacenamiento en Presa del Elefante Butte",
                  "varUnit": "Acre-Pies"
                 }]
      }]
}
```

Listing 2. SWIM workflow output excerpt for the case study.

Additional model inputs that are not included in the SWIM workflow request payload and are required in the model are assigned the default values stored in the modeling database; this feature prevents data input redundancy. The default data exclusion rule (*excludeDefault*) is used in this case study to prevent the use of default parameter values when an output from one model is to be used as an input to another, such is the case of "evap_rat_p" in Listing 1.

Listing 1 shows an example of the two rule types, the first rule disables the use of the default value for the variable parameter "evap_rat_p". The second rule defines the variables "evap_rat_p" and "SWEvapV2_ft" as semantically equivalent.

C. SWIM Workflow Output

This section describes the content of the SWIM workflow output of the case study scenario. The workflow payload follows a custom object specification divided into three blocks: metadata, provenance, and resource. Fields contained within each block are aligned to multiple standard vocabularies specified in the SWIM Vocabulary [24] in JSON-LD format. Listing 2 shows an excerpt of the SWIM workflow output. The first block of the output payload contains general metadata regarding the execution of the overall workflow; the excerpt in Listing 2 includes the execution status and the object type. Fields in this block align with Dublin Core Metadata Terms[25] in the SWIM Vocabulary; the reference to the vocabulary is included in the @context field of the JSON-LD payload.

The provenance block shows a trace of where the workflow entities originated from. For example, the entity "Model Output" with id "EBStorageV2_af" was generated by another entity with id "1fb918b3-bf35-40f3-9821-b4d907cd610f". We anticipate leveraging metadata to trace back the parent entity, in this case a modeling service, with the use of an RDF graph. Fields in this block align with the W3C PROV Namespace[26].

Finally, the resource block contains the target data elements requested by the user (for simplicity, the excerpt contains only one output data element). The data elements include metadata using the SWIM data model schema [45] with field alignments

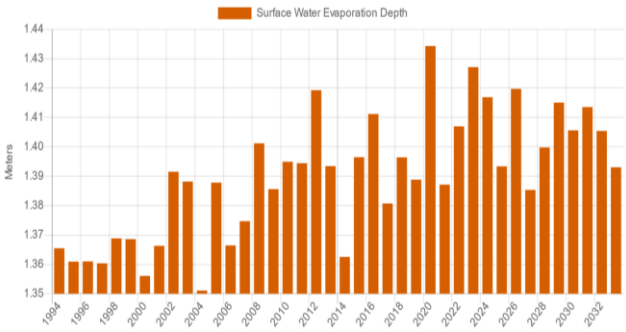

Fig. 4. SWIM screenhot of the visualization of Surface Water Evaporation Depth output values of the WBM that are sent to the HEM as Reservoir Evaporation Rate.

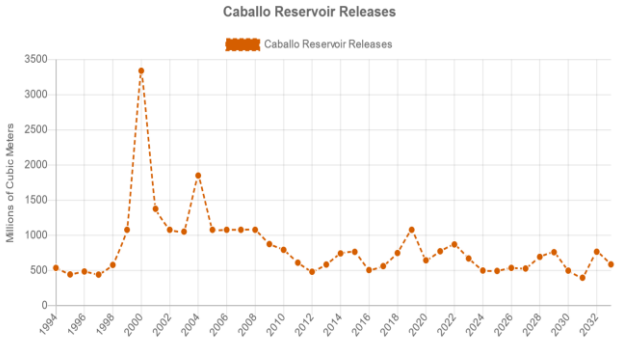

Fig. 5. SWIM screenshot of the visualization of Caballo Reservoir Releases projected by the WBM.

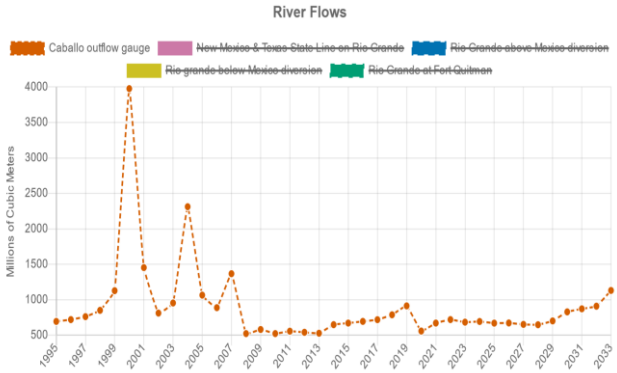

Fig. 6. SWIM screenshot of the visualization of Caballo Outflows projected by the HEM. This time series is a subset of the River Flows output which includes annual flows along different locations on the Rio Grande River.

to the SIO Ontology[27]. The generated data values from the model execution are contained within the "varValue" field.

D. Evaluation

Validating the integration of models is more dependent on the compatibility of the models, their inputs, and integration methods than on the usability of a workflow management tool. This task should consider how scientists and decision-makers can interpret the results from such complex data and model integrations. This subsection presents our initial efforts for evaluating model-to-model integration.

The SWIM orchestration infrastructure allows to seamlessly integrate the two available models by using previously described rule objects that define which models' inputs and outputs are semantically equivalent. The equivalence rule drives the automated process to generate and execute a modeling workflow. In this case study, the reservoir evaporation rate, an output of the WBM, is defined as an input to the HEM. Using SWIM's infrastructure, annual reservoir evaporation rates from the WBM are generated and used as input to the HEM. The generated model scenario (Fig. 3) for the HEM indicates that the output reservoir evaporation rate from the WBM was consumed

[24] http://purl.org/swim/vocab
[25] http://purl.org/dc/elements/1.1/
[26] https://www.w3.org/ns/prov
[27] https://bioportal.bioontology.org/ontologies/SIO

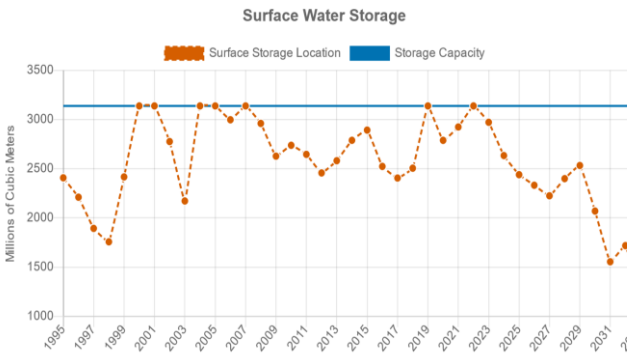

Fig. 7. SWIM screenshot of the visualization of Surface Water Storage projected by HEM. This output is a sum of the two regional reservoirs, Elephant Butte and El Caballo.

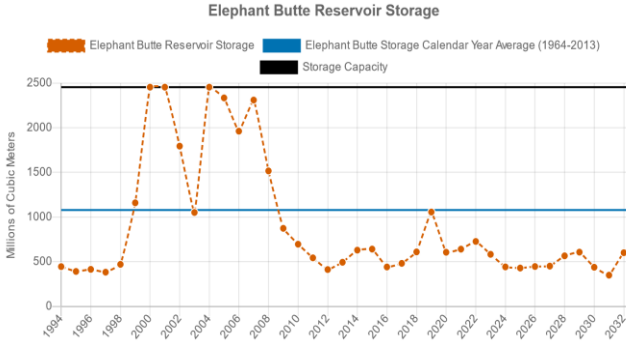

Fig. 8. SWIM screenshot of the visualization of Elephant Butte Reservoir Storage projected by the WBM.

as an input to the HEM. Fig. 4 was generated in SWIM for visualizing the Surface Water Evaporation Depth values generated by the WBM. These values are sent to the HEM as Reservoir Evaporation Rate values within the workflow process.

Being the WBM a simulation model, and the HEM an optimization model with the same area of coverage, we leveraged the SWIM infrastructure to perform a cross-validation exercise under the same scenario conditions. Conditions include the projection time range, reservoir operation rules and input water flows derived from climate projections. By restricting the optimization "wiggle room" (i.e., using same starting conditions and water distribution ranges) of the HEM within these constraints, a user can expect that equivalent outputs in both models follow similar trends.

For this evaluation, the projected outflows from El Caballo Reservoir and water storage volume upstream at Elephant Butte Reservoir were compared. Results show that Caballo outflow values "CabReleaseV2_af" projected in the WBM (Fig. 5) have a similar trend to those projected by the HEM "river_flows" (Fig. 6). This cross-validation exercise indicated compatibility of both models regarding projections of water flows at the El Caballo Reservoir location.

The reservoir storage through time in both models was also compared, and different results were observed - higher reservoir storage volume was generated in the HEM "water_stocks" than in the WBM "EBStorageV2_af" (Fig. 7 and Fig. 8). This result could be interpreted as inconsistent from a scientific perspective. Although the technical infrastructure worked as envisioned, it seems that some of the scientific model assumptions were different and were not considered when creating this scenario. One possible explanation for these results is that the input parameters for starting reservoir storage may have impacted storage through time. This result requires further discussion with domain experts/model providers to identify other differences in starting conditions or assumptions in both models that can identify scenarios for which these models can or cannot be integrated from the scientific perspective.

VII. Discussion

The need to integrate models to address complex problems in more holistic ways is recognized in [4]. The case study presented in this manuscript illustrated the use of SWIM to integrate two models with water balance capabilities (i.e., model-to-model integration) with an automated multivariable workflow composition. An important lesson learned from the case study's evaluation is that model integration needs to be validated from both the technical and the scientific perspectives. Similar challenges have been identified in the literature. Voinov and Shugart stated that attempts to integrate multiple models to achieve a more holistic understanding can quickly lead to "integronsters" [50], integrated models that generate monstrous results due to incompatible assumptions, parameters, scales, and model formulations. These incompatibilities are extremely difficult to anticipate, unless domain expertise of the integrated models is available (e.g., through a modeler or group of modelers) when scenarios that integrate models are created. In addition, since model integration aims to generate and explore more holistic models across disciplines, it remains a challenge to make this informed assessment in advance for all possible scenarios beyond checking some obvious potential incompatibilities such as spatial and temporal scale.

The SWIM infrastructure alleviates the burden of performing some manual tasks by automating: i) loading of extensive data for model consumption, ii) model-to-model workflow composition, and iii) serialization and execution of the model-to-model workflow to generate outputs of interest. SWIM users also benefit from generating data visualizations in a Web-based interface (Figs. 4-8). These features assist in comparing scenario results and identifying potential inconsistencies that could be otherwise harder to find or overseen when running the models separately. Possible solutions to validate model-to-model integration follows.

First, creating explicit mappings of common inputs to models that will be integrated (i.e., not just the output to input connections) would provide a mechanism to automatically check that these values are consistent. We believe this approach can be applied in scenarios similar to our case study to identify up-front possible inconsistencies.

Second, there is an opportunity to develop methods that scientifically validate the results of model-to-model integration. Current validation approaches focus on single models, comprehensive approaches that systematically validate the results from model integration are still to be developed. In our case study, a comparison of the graphical visualizations of

selected outputs assisted in identifying potential inconsistencies. This was an ad-hoc validation rather than a systematic approach.

Third, semantic approaches for data and model integration have been investigated [34], [40]. These approaches depend on considerable community effort in developing formal ontology(ies) and annotating the inputs and outputs of specific datasets and models using these ontologies; thus, providing domain expertise to enable the system to provide automatic checking of compatibilities and potentially automated transformations [34], [51], [52]. These efforts can also support a more comprehensive description of data and models that can be asssist users in reusing and/or repurposing data and models considering their original assumptions and potential limitations.

These initial results seem to challenge the premise of this research – that technologies can foster the use of scientific models by non-experts. Current infrastructure can certainly do this, but with the use of automation, additional tools and techniques are needed to support the interpretation of model results and under which scenarios can models be integrated and, thus, prevent model misuse, misunderstanding of model limitations, and misinterpretation of results. We discussed some solutions to prevent these issues but also recognize that this is also a human challenge, beyond the scope of this manuscript. Additional research is being conducted to investigate how users understand and use the results of these complex integrated systems by social and environmental scientists that are part of the SWIM research team. Analysis of workshops with stakeholders is currently underway to investigate how stakeholders reason with data and models of future water resources under different climate scenarios, and stakeholder perception of the usefulness of SWIM for understanding the water resource system.

VIII. Conclusions and Future Work

Computational workflows are widely used in scientific research for executing several computational processes. The use of workflows provides many advantages. However, the design and reuse of workflows is sometimes a cumbersome task if workflows are manually composed and require domain expertise. The presented approach addresses this issue with an automated workflow composer implemented as a planning microservice in SWIM that leverages Web-based technologies. The workflow composer enables the automatic composition of multivariable workflows.

The workflow composer currently implements a breadth-first, uninformed search to explore computational processes that produce target variables and returns a workflow (if possible). The accompanying infrastructure for the automatic composition of workflows provides a decoupled and abstract design of microservices that can be reused in other application domains. The proposed multivariable workflow composition algorithm can be refined by implementing a heuristic function for evaluating and selecting computational processes; thus, implementing an informed search (AI-planning). We envision a refined implementation of this algorithm as part of the ecosystem of SWIM services that enable the automated creation of scientific workflows.

The current implementation of the workflow composer in SWIM was initially evaluated in the water sustainability domain, with a case study that required integrating the HEM and WBM models. The validation of the model-to-model integration with the case study showed expected results with respect to the functionality of SWIM infrastructure. However, from the scientific perspective, potential inconsistencies in the model assumptions or conditions used in the scenario for the case study scenario were identified. This presents an opportunity for future work in the SWIM model-orchestration service pool to automatically verify the alignment of inputs and outputs of models considering scientific constraints. We anticipate using rich metadata annotations (i.e., semantics) to further describe data elements and model assumptions that affect computational processes. The use of semantics and formal requirement descriptions has been previously explored in [34] and [42]. We also discussed the challenges of validating the integration of models from the scientific perspective, since models are being used beyond their original purpose, and possible approaches to address those challenges and prevent unexpected use of these frameworks, such as misinterpretation of results.

In addition to verification of data alignment, metadata annotations can be leveraged for the automated generation of rules for model-to-model integration (e.g., equivalence rules) introduced in section VI.B. These efforts can support the work of domain experts in aligning scientific variables across models.

SWIM's workflow composer creates a provenance trace of the resulting workflow. Our future work includes the generation of an RDF graph for representing and enriching this information. Our current implementation does not allow users to control the granularity of the workflow provenance trace. We anticipate that different levels of execution diagnostic data can be leveraged from WMSs' logs.

Efforts towards automating model-to-model integration with consistency validation both from the technical and scientific perspectives can support this task but still require domain expertise; thus, they need to leverage both human and machine capabilities. These efforts can significantly support scientific endeavors and decision-making by enabling a wide variety of stakeholders to focus on the use of scientific models instead of using decoupled modules that require the manual curation of data and use of various tools and infrastructure.

Acknowledgment

The authors wish to thank the research team and collaborators (scientists and students) that have participated in SWIM for their invaluable contributions. A special thanks goes to Bill Hargrove, Alfredo Granados, Frank Ward, Alex Mayer, and Dave Gutzler, as well as the anonymous reviewers for their insightful comments for the improvement of this manuscript.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

REFERENCES

[1] "Review of Targets for the Sustainable Development Goals: The Science Perspective (2015)," International Science Council. https://council.science/publications/review-of-targets-for-the-sustainable-development-goals-the-science-perspective-2015/ (accessed Aug. 17, 2022).

[2] S. H. Hamilton, S. ElSawah, J. H. A. Guillaume, A. J. Jakeman, and S. A. Pierce, "Integrated assessment and modelling: Overview and synthesis of salient dimensions," Environmental Modelling & Software, vol. 64, pp. 215–229, Feb. 2015, doi: 10.1016/j.envsoft.2014.12.005.

[3] G. F. Laniak et al., "Integrated environmental modeling: A vision and roadmap for the future," Environmental Modelling & Software, vol. 39, pp. 3–23, Jan. 2013, doi: 10.1016/j.envsoft.2012.09.006.

[4] G. F. Belete, A. Voinov, and G. F. Laniak, "An overview of the model integration process: From pre-integration assessment to testing," Environmental Modelling & Software, vol. 87, pp. 49–63, Jan. 2017, doi: 10.1016/j.envsoft.2016.10.013.

[5] J. Carrillo, D. Garijo, M. Crowley, R. Carrillo, Y. Gil, and K. Borda, "Semantic Workflows and Machine Learning for the Assessment of Carbon Storage by Urban Trees," presented at the Proceedings of the 3rd. International Workshop on Capturing Scientific Knowledge collocated with the 10th. International Conference on Knowledge Capture (K-CAP '19), Los Angeles, CA, Nov. 2019. [Online]. Available: http://ceur-ws.org/Vol-2526/paper1.pdf

[6] E. Deelman et al., "The future of scientific workflows," The International Journal of High Performance Computing Applications, vol. 32, no. 1, pp. 159–175, Jan. 2018, doi: 10.1177/1094342017704893.

[7] Y. Gil et al., "Examining the challenges of scientific workflows," Computer, vol. 40, no. 12, pp. 24–32, 2007, doi: 10.1109/MC.2007.421.

[8] M. R. Crusoe et al., "Methods Included: Standardizing Computational Reuse and Portability with the Common Workflow Language," Commun. ACM, vol. 65, no. 6, pp. 54–63, May 2022, doi: 10.1145/3486897.

[9] B. Jennings and R. Stadler, "Resource Management in Clouds: Survey and Research Challenges," J Netw Syst Manage, vol. 23, no. 3, pp. 567–619, Jul. 2015, doi: 10.1007/s10922-014-9307-7.

[10] K. Burkat et al., "Serverless Containers – Rising Viable Approach to Scientific Workflows," in 2021 IEEE 17th International Conference on eScience (eScience), Sep. 2021, pp. 40–49. doi: 10.1109/eScience51609.2021.00014.

[11] M. D. Wilkinson et al., "The FAIR Guiding Principles for scientific data management and stewardship," Sci Data, vol. 3, no. 1, p. 160018, Mar. 2016, doi: 10.1038/sdata.2016.18.

[12] A. Zia et al., "Coupled impacts of climate and land use change across a river–lake continuum: insights from an integrated assessment model of Lake Champlain's Missisquoi Basin, 2000–2040," Environ. Res. Lett., vol. 11, no. 11, p. 114026, Nov. 2016, doi: 10.1088/1748-9326/11/11/114026.

[13] K. Vahi et al., "Custom Execution Environments with Containers in Pegasus-Enabled Scientific Workflows," in 2019 15th International Conference on eScience (eScience), Sep. 2019, pp. 281–290. doi: 10.1109/eScience.2019.00039.

[14] N. Pavlovikj, K. Begcy, S. Behera, M. Campbell, H. Walia, and J. S. Deogun, "A Comparison of a Campus Cluster and Open Science Grid Platforms for Protein-Guided Assembly Using Pegasus Workflow Management System," in 2014 IEEE International Parallel Distributed Processing Symposium Workshops, May 2014, pp. 546–555. doi: 10.1109/IPDPSW.2014.66.

[15] B. Riedel et al., "Distributed Data and Job Management for the XENON1T Experiment," in Proceedings of the Practice and Experience on Advanced Research Computing, New York, NY, USA, Jul. 2018, pp. 1–8. doi: 10.1145/3219104.3219155.

[16] P. Maechling et al., "Simplifying construction of complex workflows for non-expert users of the southern california earthquake center community modeling environment," ACM SIGMOD Record, vol. 34, no. 3, pp. 24–30, 2005, doi: 10.1145/1084805.1084811.

[17] G. S. Davies, T. Dent, M. Tápai, I. Harry, C. McIsaac, and A. H. Nitz, "Extending the PyCBC search for gravitational waves from compact binary mergers to a global network," Phys. Rev. D, vol. 102, no. 2, p. 022004, Jul. 2020, doi: 10.1103/PhysRevD.102.022004.

[18] M. Kotliar, A. V. Kartashov, and A. Barski, "CWL-Airflow: a lightweight pipeline manager supporting Common Workflow Language," GigaScience, vol. 8, no. 7, p. giz084, Jul. 2019, doi: 10.1093/gigascience/giz084.

[19] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple Linux Utility for Resource Management," in Job Scheduling Strategies for Parallel Processing, Berlin, Heidelberg, 2003, pp. 44–60. doi: 10.1007/10968987_3.

[20] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience," Concurrency and Computation: Practice and Experience, vol. 17, no. 2–4, pp. 323–356, 2005, doi: 10.1002/cpe.938.

[21] J. J. Rehr, F. D. Vila, J. P. Gardner, L. Svec, and M. Prange, "Scientific Computing in the Cloud," Computing in Science Engineering, vol. 12, no. 3, pp. 34–43, May 2010, doi: 10.1109/MCSE.2010.70.

[22] G. Juve and E. Deelman, "Scientific Workflows in the Cloud," in Grids, Clouds and Virtualization, M. Cafaro and G. Aloisio, Eds. London: Springer, 2011, pp. 71–91. doi: 10.1007/978-0-85729-049-6_4.

[23] E. Deelman et al., "The Evolution of the Pegasus Workflow Management Software," Computing in Science & Engineering, vol. PP, pp. 1–1, May 2019, doi: 10.1109/MCSE.2019.2919690.

[24] Y. Zhao, X. Fei, I. Raicu, and S. Lu, "Opportunities and Challenges in Running Scientific Workflows on the Cloud," in 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Oct. 2011, pp. 455–462. doi: 10.1109/CyberC.2011.80.

[25] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The Journey So Far and Challenges Ahead," IEEE Software, vol. 35, no. 3, pp. 24–35, May 2018, doi: 10.1109/MS.2018.2141039.

[26] I. Salvadori, A. Huf, R. dos S. Mello, and F. Siqueira, "Publishing Linked Data Through Semantic Microservices Composition," in Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services, New York, NY, USA, 2016, pp. 443–452. doi: 10.1145/3011141.3011155.

[27] T. Šimko, L. Heinrich, H. Hirvonsalo, D. Kousidis, and D. Rodríguez, "REANA: A System for Reusable Research Data Analyses," EPJ Web Conf., vol. 214, p. 06034, 2019, doi: 10.1051/epjconf/201921406034.

[28] R. Mačiulaitis et al., "Support for HTCondor high-Throughput Computing Workflows in the REANA Reusable Analysis Platform," in 2019 15th International Conference on eScience (eScience), Sep. 2019, pp. 630–631. doi: 10.1109/eScience.2019.00091.

[29] D. Godlove, "Singularity: Simple, secure containers for compute-driven workloads," in Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning), New York, NY, USA, Jul. 2019, pp. 1–4. doi: 10.1145/3332186.3332192.

[30] L. Gerhardt et al., "Shifter: Containers for HPC," J. Phys.: Conf. Ser., vol. 898, p. 082021, Oct. 2017, doi: 10.1088/1742-6596/898/8/082021.

[31] N. Villanueva-Rosales, N. del Rio, D. Pennington, and L. Garnica Chavira, "Semantic Bridges for Biodiversity Sciences," in The Semantic Web - ISWC 2015, Cham, 2015, pp. 310–317. doi: 10.1007/978-3-319-25010-6_20.

[32] M. D. Wilkinson, B. Vandervalk, and L. McCarthy, "The Semantic Automated Discovery and Integration (SADI) Web service Design-Pattern, API and Reference Implementation," Journal of Biomedical Semantics, vol. 2, no. 1, p. 8, Oct. 2011, doi: 10.1186/2041-1480-2-8.

[33] N. Del Rio, N. Villanueva-Rosales, D. Pennington, K. Benedict, A. Stewart, and C. J. Grady, "Elseweb meets sadi: Supporting data-to-model integration for biodiversity forecasting," 2013. [Online]. Available: https://www.aaai.org/ocs/index.php/FSS/FSS13/paper/view/7631

[34] Y. Gil et al., "Wings: Intelligent Workflow-Based Design of Computational Experiments," IEEE Intelligent Systems, vol. 26, no. 1, pp. 62–72, Jan. 2011, doi: 10.1109/MIS.2010.9.

[35] S. Peckham, "The CSDMS Standard Names: Cross-Domain Naming Conventions for Describing Process Models, Data Sets and Their Associated Variables," International Congress on Environmental Modelling and Software, Jun. 2014, [Online]. Available: https://scholarsarchive.byu.edu/iemssconference/2014/Stream-A/12

[36] R. Ferreira da Silva, D. Garijo, S. Peckham, Y. Gil, E. Deelman, and V. Ratnakar, “Towards Model Integration via Abductive Workflow Composition and Multi-Method Scalable Model Execution,” International Congress on Environmental Modelling and Software, Jun. 2018, [Online]. Available: https://scholarsarchive.byu.edu/iemssconference/2018/Stream-A/14

[37] D. Garijo et al., “A Semantic Model Catalog to Support Comparison and Reuse,” International Congress on Environmental Modelling and Software, Jun. 2018, [Online]. Available: https://scholarsarchive.byu.edu/iemssconference/2018/Stream-A/11

[38] S. Gupta, P. Szekely, C. A. Knoblock, A. Goel, M. Taheriyan, and M. Muslea, “Karma: A System for Mapping Structured Sources into the Semantic Web,” in The Semantic Web: ESWC 2012 Satellite Events, Berlin, Heidelberg, 2015, pp. 430–434. doi: 10.1007/978-3-662-46641-4_40.

[39] A. Karpatne, Z. Jiang, R. R. Vatsavai, S. Shekhar, and V. Kumar, “Monitoring Land-Cover Changes: A Machine-Learning Perspective,” IEEE Geoscience and Remote Sensing Magazine, vol. 4, no. 2, pp. 8–21, Jun. 2016, doi: 10.1109/MGRS.2016.2528038.

[40] Y. Gil et al., “MINT: model integration through knowledge-powered data and process composition,” in 9th International Congress on Environmental Modelling and Software, 2018, vol. 8. [Online]. Available: https://scholarsarchive.byu.edu/iemssconference/2018/Stream-A/13/

[41] Y. Gil et al., “Artificial Intelligence for Modeling Complex Systems: Taming the Complexity of Expert Models to Improve Decision Making,” ACM Trans. Interact. Intell. Syst., vol. 11, no. 2, p. 11:1-11:49, Jul. 2021, doi: 10.1145/3453172.

[42] V. Kasalica and A.-L. Lamprecht, “Workflow Discovery with Semantic Constraints: The SAT-Based Implementation of APE,” Electronic Communications of the EASST, vol. 78, no. 0, Art. no. 0, May 2020, doi: 10.14279/tuj.eceasst.78.1092.

[43] I. A. Klampanos et al., “DARE Platform\: a Developer-Friendly and Self-Optimising Workflows-as-a-Service Framework for e-Science on the Cloud,” Journal of Open Source Software, vol. 5, no. 54, p. 2664, Oct. 2020, doi: 10.21105/joss.02664.

[44] R. Filgueira, A. Krause, M. Atkinson, I. Klampanos, A. Spinuso, and S. Sanchez-Exposito, “dispel4py: An Agile Framework for Data-Intensive eScience,” in 2015 IEEE 11th International Conference on e-Science, Aug. 2015, pp. 454–464. doi: 10.1109/eScience.2015.40.

[45] L. Garnica Chavira, J. Caballero, N. Villanueva-Rosales, and D. Pennington, “Semi-structured Knowledge Models and Web Service Driven Integration for Online Execution and Sharing of Water Sustainability Models,” International Congress on Environmental Modelling and Software, Jun. 2018, [Online]. Available: https://scholarsarchive.byu.edu/iemssconference/2018/Stream-A/43

[46] S. Rusell and P. Norvig, Artificial Intelligence: A Modern Approach, 1st Edition. United States of America: Prentice-Hall, Inc., 1995.

[47] P. Amstutz et al., “Common Workflow Language, v1.0,” 2016, doi: https://doi.org/10.6084/m9.figshare.3115156.v2.

[48] R. N. Holmes, A. Mayer, D. S. Gutzler, and L. G. Chavira, “Assessing the Effects of Climate Change on Middle Rio Grande Surface Water Supplies Using a Simple Water Balance Reservoir Model,” Earth Interactions, vol. 26, no. 1, pp. 168–179, Jan. 2022, doi: 10.1175/EI-D-21-0025.1.

[49] F. A. Ward, A. S. Mayer, L. A. Garnica, N. T. Townsend, and D. S. Gutzler, “The economics of aquifer protection plans under climate water stress: New insights from hydroeconomic modeling,” Journal of Hydrology, vol. 576, pp. 667–684, Sep. 2019, doi: 10.1016/j.jhydrol.2019.06.081.

[50] A. Voinov and H. H. Shugart, “‘Integronsters’, integral and integrated modeling,” Environmental Modelling & Software, vol. 39, pp. 149–158, Jan. 2013, doi: 10.1016/j.envsoft.2012.05.014.

[51] B. T. Essawy, J. L. Goodall, H. Xu, and Y. Gil, “Evaluation of the OntoSoft Ontology for describing metadata for legacy hydrologic modeling software,” Environmental Modelling & Software, vol. 92, pp. 317–329, Jun. 2017, doi: 10.1016/j.envsoft.2017.01.024.

[52] M. Stoica and S. D. Peckham, “An Ontology Blueprint for Constructing Qualitative and Quantitative Scientific Variables.,” 2018.