Designing for Real-Time Groupware Systems to Support Complex Scientific Data Analysis

GOLAM MOSTAEEN, University of Saskatchewan, Canada
BANANI ROY, University of Saskatchewan, Canada
CHANCHAL K. ROY, University of Saskatchewan, Canada
KEVIN A. SCHNEIDER, University of Saskatchewan, Canada

Scientific Workflow Management Systems (SWfMSs) have become popular in recent years for accelerating the specification, execution, visualization, and monitoring of data-intensive tasks. Unfortunately, to the best of our knowledge no existing SWfMSs directly support collaboration. Data is increasing in complexity, dimensionality, and volume, and the efficient analysis of data often goes beyond the realm of an individual and requires collaboration with multiple researchers from varying domains. In this paper, we propose a groupware system architecture for data analysis that in addition to supporting collaboration, also incorporates features from SWfMSs to support modern data analysis processes. As a proof of concept for the proposed architecture we developed *SciWorCS* - a groupware system for scientific data analysis. We present two real-world use-cases: collaborative software repository analysis and bioinformatics data analysis. The results of the experiments evaluating the proposed system are promising. Our bioinformatics user study demonstrates that SciWorCS can leverage real-world data analysis tasks by supporting real-time collaboration among users.

Additional Key Words and Phrases: Scientific; Data; Analysis; Workflow; Collaboration

ACM Reference Format:
Golam Mostaeen, Banani Roy, Chanchal K. Roy, and Kevin A. Schneider. 2019. Designing for Real-Time Groupware Systems to Support Complex Scientific Data Analysis. *Proc. ACM Hum.-Comput. Interact.* 3, EICS, Article 09 (June 2019), 28 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In the recent big data era, scientific experiments need to handle massive amounts of heterogeneous data [43, 47]. While data-intensive experiments open up the possibilities for interesting discoveries (such as through statistical analysis and applying advanced machine learning algorithms), there are several challenges with complex, data-intensive computation and the analysis process, such as dealing with failure handling, optimal task scheduling, big data visualization, distributed job execution and real time execution monitoring [43, 46]. The effective analysis and management of complex, multi-dimensional, and high volume data is challenging for an individual and often requires collaboration with multiple scientists [87, 88]. In any case, scientific research often demands collaboration among scientists from multiple domains and with diverse expertise [46, 73, 74, 87, 88].

Authors' addresses: Golam Mostaeen, University of Saskatchewan, Saskatoon, SK, Canada, golam.mostaeen@usask.ca; Banani Roy, University of Saskatchewan, Saskatoon, SK, Canada, banani.royn@usask.ca; Chanchal K. Roy, University of Saskatchewan, Saskatoon, SK, Canada, chanchal.roy@usask.ca; Kevin A. Schneider, University of Saskatchewan, Saskatoon, SK, Canada, kevin.schneider@usask.ca.

2573-0142/2019/6-ART09 $15.00
https://doi.org/10.1145/1122445.1122456

Given the collaborative nature of modern complex scientific experiments, recent research determined that Computer-Supported Collaborative Work (CSCW) technologies are necessary to support scientific experiments that require collaboration among multiple researchers [13, 27, 34, 88]. Jirotka et al. determined that, while the relationship between scientific experiments (i.e., *e-Science*) and CSCW is relatively nascent, they together could yield significant benefits in answering complex research questions and in important knowledge discoveries [34]. With this motivation, we studied the concept of collaboration or groupware systems in the context of *Scientific Workflow Management Systems (SWfMSs)* for aiding with complex scientific experiments among multiple scientists via real-time collaboration. In this paper, we present several challenges and differences with collaborative SWfMSs in contrast to text or graphics editing groupware systems, and present a general framework for collaborative SWfMSs that leverages CSCW technologies to support scientific experiments.

A scientific workflow is a facilitation or automation of a process as a part or whole [3, 33] during which the targeted data are passed from one computational step to another for certain actions or processing as per some set of pre-defined rules or instructions [3, 33, 43]. A SWfMS automates a scientific workflow life cycle: composition, deployment, execution, and analysis [43, 47], which is discussed in detail in Section 2. While SWfMSs are widely used in recent years for handling and managing the overall execution of complex scientific experiments [43, 46], none of them directly support collaborative work among multiple users; hence users need to follow several time consuming manual steps for any required collaboration on a given data analysis task [73, 74, 87, 88]. For example, for a collaborative design of a scientific workflow, a user first builds a part of a workflow (e.g., a *sub-workflow*), exports it from the local workflow engine and shares it with a collaborator for possible updates on the sub-workflow. Around 3910 such scientific workflows have been shared among 10665 members (as last noted in August 2018) for collaboration in *myExperiment*[14] — a shared social space for scientific artifacts. The manual collaboration process is repeated a number of times to complete building the entire workflow comprising of several sub-workflows. This manual back and forth process for collaboration is often very time consuming, does not support real-time editing, and is often impractical as the collaborating group size increases over time.

While the above statistics and scenario reveals the importance and necessity of collaborative SWfMSs, designing a real-time groupware system for workflow collaboration is non-trivial and differs from text or graphics editing groupware systems in a number of ways: i) *Different Roles.* Scientific experiments often require adequate access control policies for sharing the workflow components, data products and provenance information among researchers with varying roles [5]. In the context of scientific data analysis, the varying roles might include: Domain user, Pipeline composer, tool developer or Data specialist and so on depending on the given use-case scenarios [46]. We present further details on varying roles in Section 5.3. ii) *Collaborative Job Scheduling.* Collaborative job scheduling is required to orchestrate and efficiently schedule the independent workflow execution requests of researchers [87, 88], iii) *Collaborative Job Management.* In addition to the primary requirement of workflow job execution, monitoring or failure handling, collaborative SWfMSs need to have a feedback system to orchestrate the overall data analysis process among the collaborators, iv) *Plugin Architecture for Collaboration.* For effective collaboration among research groups, a collaborative SWfMS must allow easy and real-time plugins of workflow tools, and v) *Collaborative Data Visualization.* A collaborative SWfMS should facilitate collaborative data visualization to fully exploit collaborative data analysis.

To address these challenges and requirements we present a framework towards an effective design of a collaborative SWfMS for scientific data analysis. Our proposed framework adopts a plugin based architecture for workflow tools. As a proof of concept of the proposed framework, we also implement a collaborative SWfMS — *SciWorCS*. As our proposed framework is not restricted to

any particular research domain, we evaluate it with use-cases from two different research areas — *Bioinformatics* and *Software Repository Analysis* — where the framework demonstrates promising results and significant potential.

The main contributions of our work are as follows:

(1) We carefully identify the challenges and requirements for a collaborative data analysis platform.
(2) We propose a general framework for collaborative data analysis which is applicable to multiple modern research domains.
(3) We present SciWorCS[1] — a collaborative SWfMS, developed as a proof of concept of the proposed framework.
(4) We present two case studies using SciWorCS in the domains of *Bioinformatics* and *Software Repository Analysis*.
(5) We show how users perform collaborative tasks using SciWorCS in the domain of Bioinformatics.

Outline. The rest of the paper is organized as follows. We first present technical prerequisites and background in Section 2. We discuss related existing research work in Section 3. In Section 4, we discuss the identified requirement analysis and formulate the problems towards the design of collaborative SWfMSs. We then present our proposed framework in Section 5. We discuss the implementation details and technical features of SciWorCS as a proof of concept of the proposed framework with different evaluations in Section 6 and the results are discussed in Section 7. Section 8 discusses the possible threats to validity and finally we draw conclusions in Section 9.

2 BACKGROUND

In this section, we discuss *Scientific Workflows* and their general life cycle followed by a general overview of *Collaborative Data Analysis*.

2.1 Scientific Workflows and Data Analysis

A workflow is a facilitation or automation of a process as a part or whole [3, 33] during which the targeted data are passed from one participant (e.g., human or computer) to another for certain actions or processing as per some set of rules [3, 33, 43]. *Scientific workflows* operate at an abstract level, and are used for modeling and performing scientific experiments on a dataset [3, 43]. I0n terms of scientific workflows, the workflow steps are more commonly referred to as *Computational Modules*. A computational module is responsible for some independent tasks of data manipulation and processing. The modules define the associated input data format, the data processing methods and the corresponding output data format [43].

From composition to analysis, the life cycle of a scientific workflow can broadly be categorized into four phases [15, 28, 43]: i) *Composition Phase* [15, 43, 51], in this phase the creation of the workflow comprising the modular abstraction is done; ii) *Deployment Phase* [15, 43], while the composition phase defines the abstract level of the workflow, in the deployment phase the composed workflow is prepared for execution with the required setups; iii) *Execution Phase*, the deployed workflow is executed with the associated input data in this phase to produce the corresponding output of the workflow [15, 43, 51]; and, iv) *Analysis Phase*, finally in this phase the output data is analyzed as per some research hypotheses. Workflow data provenance, output data visualization and so on are some examples of the Analysis phase [43].

[1] https://github.com/blindedForReviewPhase

2.2 Collaborative Data Analysis

The generation of large amounts of heterogeneous data on a daily basis by different areas of modern science has influenced many disciplines in moving towards data and information driven analysis and discoveries [47]. The volume or complexity of the data often requires collaborative analysis involving multiple researchers. For example, in a similar study to ours Zhang et al. [88] referred to the *Large Synoptic Survey Telescope (LSST)* [45] experiment that demands a collaboration of around 1800 scientists and engineers for a complete analysis process. As well, some scientific domains essentially require collaboration as they are highly connected to multiple research disciplines [87, 88], such as the Plant Phenotyping and Genotyping research domain [54]. Hence, the notion of collaborative data analysis was coined where multiple researchers work on a project dataset for different data manipulation, execution, analysis, visualization, or interesting knowledge discoveries [5, 46, 74, 87, 88].

Table 1. Proposed Architecture Vs. Existing Research Work.

#	Research Work	CM [1]	AC [2]	C. Viz. [3]	C. JQ [4]	C. Prov. [5]	C. UA [6]
1	Zhang [87]	✓	✗	✗	✗	✗	✓
2	Lu et al. [46]	✓	✗	✗	✓	✗	✗
3	Bhuyan et al. [5]	✗	✓	✗	✗	✗	✗
4	Sipos et al. [73–76]	✓	✗	✗	✗	✗	✗
5	*Galaxy* [25], *Taverna* [62], *Kepler* [47], *Pegasus* [16]	✗	✓	✗	✗	✗	✗
6	Proposed Architecture	✓	✓	✓	✓	✓	✓

1: Consistency Management, 2: Access Control 3: Collaborative Visualization
4: Collaborative Job Queuing, 5: Collaborative Provenance, 6: Collaborative User Awareness

3 RELATED WORK

In this section, we first present related work on CSCW in aiding with scientific experiments (i.e., in Section 3.1), and we then discuss recent work for supporting collaborative data analysis with SWfMSs (i.e., in Section 3.2).

3.1 CSCW to Support Scientific Experiments

Several recent researches assert the necessity of CSCW in supporting complex scientific experiments that require collaboration among multiple researchers [13, 27, 34, 88]. Jirotka et al. from their investigation studies presented that, while the relation of scientific experiments (i.e., e-Science) and CSCW are relatively nascent one, they exhibit significant potentials in answering complex research questions and in important knowledge discovery [34]. Hence, over the past few years several research studies have been conducted in understanding human behavior [21, 27]. Besides some of the study targeted on discrete aspects of collaborative data analysis systems, such as consistency management or locking schemes [34, 88]. However, to the best of our knowledge none of the previous study addressed the overall architecture comprising different primary requirement or components for a collaborative data analysis system. Hence, our study and proposed architecture is different than the existing studies in the sense that we formulate the discrete research problems towards designing an overall architecture towards collaborative data analysis.

A number of studies have been conducted in recent years for gaining in depth understanding

of scientific work practices such as, how scientific experiments are conducted, how research artifacts are shared, how scientists interact for tools and technologies and so on [34, 48, 60, 67]. While such investigations often target divergent scientific experiments (e.g., the Electronic Medical Record (EMR) [32], Breast Cancer Screening [35] and so on), they generally aim in providing important insights on challenges and design implications of CSCW systems towards virtual workspace for collaborative scientific experiments [34, 48]. As we propose a high-level architecture for collaborative data analysis platform, the insights from the existing studies exhibit the scope of integrating with our proposed architecture towards designing a domain specific collaborative data analysis platform, such as for EMR, Breast Cancer Screening and so on.

3.2 Towards Collaborative Data Analysis

A Scientific Workflow Management System (SWfMS) automates the process of life cycle phases-composition, deployment, execution and analysis of a scientific workflow [43, 47]. Large-scale scientific experiments often take advantages of SWfMSs for modeling the overall data analysis and manipulation process comprising of different computational steps for input data loading, transformation, aggregation and so on [43] where, SWfMSs work as a framework for supporting the specification, modification, execution, failure handling, and monitoring of the data-intensive tasks [43, 46]. With the increase of data complexity and volume, extensive research has been done on this domain resulting a number of proposed SWfMSs architecture and corresponding implementation. Some of the modern popular SWfMSs are: *Galaxy* [25], *Taverna* [62], *Kepler* [47], *Pegasus* [16], *VisTrails* [9], *Triana* [82], *VIEW* [42], *Chiron* [61], *GridNexus* [8]. However, to the best of our knowledge none of the SWfMSs supports collaboration directly and requires manual effort in contrast to our proposed architecture supporting real-time collaboration for data analysis.

Lu *et al.* [46] studied several motivations opportunities for collaborative SWfMSs from the perspective of large-scale and multidisciplinary research projects. A number of methods have been proposed for consistency management of the shared workflow in a collaborative environment. Zhang *et al.* [87] studied the concept of turn based locking scheme in the context of collaborative SWfMSs for facilitating the consistency management. In such setup, each collaborators generally has only the *Read* access to the shared workflow. Collaborators request and compete for the floor for carrying out any update or transaction on the workflow (e.g., *Read & Write* access). Fei *et al.* [20] and Zhang *et al.* [88] presented locking schemes by allowing only descendent module locks (e.g., descendent nodes of the workflow DAG [43]).

Sipos *et al.* [73] used two lock modes - *User* and *System* locks. Fei *et al.* [20] proposed a lock compatibility matrix for a set of six pre-defined modes of locks. Besides, techniques have also been studied for extending the single-user Grid portals to a collaborative environment [74, 76].

While the locking schemes for consistency management is one of the primary requirements of a collaborative system [79], such the requirements of collaborative SWfMSs are often more than that [46]. For example, those studies did not consider several important aspects such as collaborative job scheduling, collaborative job execution, collaborative visualization, user interaction for problem solving and so on. Hence, in our study we addressed those requirements as well for a successful collaborative data analysis platform, which can significantly contribute towards further improvement in the research domain.

Table 1 demonstrates the summary of the proposed architecture in comparison to some of the existing related research. The table shows that, while existing research targeted different discrete components, our proposed architecture addresses all of the primary requirements towards the efficient design of collaborative data analysis platform.

4 RESEARCH CHALLENGES AND PROBLEM FORMULATION

In order to design a high-level architecture of a collaborative SWfMS, we first attempted to identify the challenges in this domain. We thus first studied the existing literature (see Section 3 for quick reference) and learned about the challenges. We then conducted a survey with scientists of different domains to understand researchers' requirements in scientific analysis, their current practices and encountered challenges. Finally, we analyzed the existing SWfMSs including Galaxy [25] by using their executables, source code, online documentation, videos, asking questions of the product developers, and presenting and demonstrating the frameworks to stakeholders. We also confirmed and adopted the challenges identified by Roy et al. [69] in the collaborative context which follows an adapted Scenario-based based architectural analysis method (SAAM) proposed by Roy and Graham [68]. In particular, they focus on how different stakeholders use these systems in a variety of scenarios and to what extent those systems support their needs, which we reused in our study. Hence, prior to presenting our proposed architecture in Section 5, here in this section we first discuss the primary challenges and formulate the problem that we target to solve in our proposed architecture.

4.1 Consistency Management in Collaborative Scientific Workflow Composition

One of the most important challenges for any real-time collaborative system is the consistency management of the shared objects in the face of conflicting operations by the collaborators [79, 80]. In a collaborative editing system, the concurrent operations on the same shared object might create several conflicting states at any given time frame. Generally, different version controlling techniques, such as SVN - Subversion, CVS - Concurrent Versions System and so on are widely used for conflict resolution of the unstructured document collaborative systems, such as Collaborative Text Document Editing [65], Collaborative Computer Aided Design (CAD) [12], object-based collaborative Graphics Editing systems [79], Collaborative Bitmap Editing System [23].

Unlike these documents, the scientific workflows are more structured where one module can be highly dependent on another due to dataflow relation in between them [88]. Even any minor changes in any part of a workflow, can significantly impact the other part of the collaborative workflow in execution and data manipulation [19, 20, 87, 88]- which often make the problem notably different than that of unstructured document collaborative systems, such as text or graphics editing systems [54, 74, 88]. Hence, consistency management in the face of conflicting concurrent operations in collaborative workflow composition has been one of the important research problems in this domain [19, 20, 54, 74, 87, 88].

4.2 Collaboration Modeling for Varying User Roles

Studies show that the identification and modeling of varying user roles in the context of complex scientific experiments and data analysis can often be crucial and different than that of unstructured document collaborative systems [5, 11, 49]. Majority of the scientific experiments do not suggest clear direction on how to proceed or on the correctness of the solution and thus often exhibit '*ill-defined*' problem nature [10] — where the solutions are not often exact and also there exists several paths or strategies towards the possible solution [4, 10]. Besides, a scientific workflow is comprised of and linked with different components, such as workflow module ports, executable, datalink, data products, provenance information [5, 20, 88]. Managing the usability and accessibility of the workflow components among the collaborating parties are important to guarantee their domain specific expertise matching, security of the components and easier access [5].

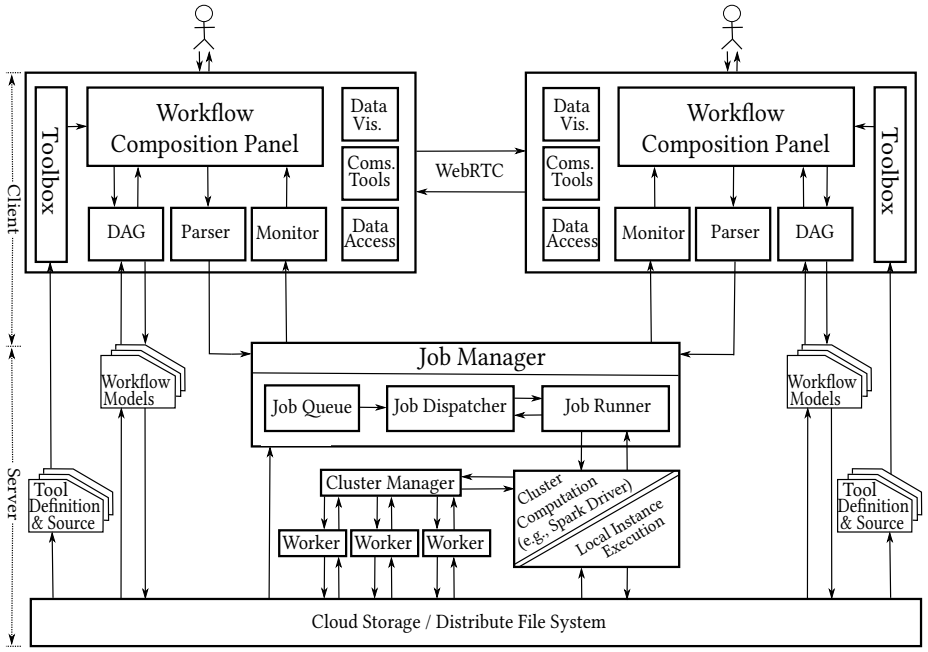

Fig. 1. High-level Architecture of a Collaborative SWfMS

4.3 Collaboration and Workflow Execution

Lu et al. investigated the possible challenges towards the execution and tasks management from collaborative perspective, such as maintaining a collaborative provenance model, the relationship between scientific workflows and collaboration models [46]. In a similar study, Zhang [87, 88] presented that often there can have several sub-groups working collaboratively on different sub-workflows of an entire scientific workflows. A collaborative SWfMS needs to handle such independent sub-workflow execution and backdoor communication among the sub-group collaborators [88]. A collaborative scientific data analysis hence, must provide facilities for queuing and handling execution plans by independent collaborative sub-groups.

In the context of scientific data analysis, provenance management is important for supporting the reproducibility of scientific experiments, result interpretation or problem diagnosis [22, 43]. The scientific experiments from the collaborative perspective also demands improved provenance management for the multi-level and heterogeneous executions [43].

4.4 Plugin Based Architecture

The data analysis tasks are generally conducted with sets of individual reusable computational units or modules [1]. The computational units are often configured as per the specific data manipulation steps [1, 47]. As different researchers often work towards smaller sub-tasks in a collaborative, the collaborative data analysis platform hence primarily need to support easier real-time pluggig-in of the computational units. The plug-in support thus allows extension and solving of more complex problems over time in collaboration.

4.5 User Interaction and Problem Solving

Research studies show that, when group activities are not properly channeled and coordinated, the interaction among the members could affect the iterative process of solution finding and thus negatively affect the solution [17]. Given the complex nature of scientific data analysis involving users of different roles, we need to model the communication and awareness among collaborators. In the context of collaborative SWfMSs, the required interaction can be of, i) awareness or communication and ii) data visualization among collaborators.

4.5.1 Awareness and Communication Among Collaborators. Providing different methodologies for group communication towards problem-solving and decision making are also very important towards the success of a groupware system [17, 40]. Given the 'ill-defined' nature of the data analysis tasks [10], effective real-time interaction among the users are often at the heart of exploiting the collaboration and problem solving [17]. Hence, similar to other groupware system, the collaborative data analysis platform need to facilitate real-time user communication (i.e., text, audio, visual and so on) and user awareness (i.e., telepointer, heatmap and so on) towards aiding problem solving.

4.5.2 Collaborative Data Visualization. In terms of scientific data analysis, data visualization is also an essential component for interesting knowledge discovery and problem solving, such as of *Genomic data analysis* [83]. The collaborative platform hence should support easy plugging-in of new data visualization tools and aid in collaborative visualization such as via visualization components sharing, real-time annotation and so on. Research studies show that, in addition to the user awareness and communication, such support for data visualization and collaborative problem solving can be important for the successful design of groupware systems [17].

5 PROPOSED ARCHITECTURE

Figure 1 illustrates the proposed high-level architecture of a collaborative SWfMS. As noticeable from the figure, the proposed architecture comprises of different components and relations among them. In this section we present the technical functions of the components and also explain how they interact with one another.

5.1 Plugin Based Architecture for Collaborative Workflow Composition

The data analysis is powered by a set of reusable computational modules, which are integrated to collaborative data analysis platform as different plugins by the API developers. As plugin-based architecture is one of the primary requirements for the extension of collaborative data analysis over time (e.g., as discussed in Section 4), in this section we discuss our proposed approach towards achieving the goal using the two primary components of the architecture — *Toolbox* and *Workflow Model*.

5.1.1 Toolbox. A workflow for data analysis is often represented as a *Directed Acyclic Graph (DAG)*, $W = (M, E)$, where M is a set of n finite number of modular tasks, $m_i, (1 \leq i \leq n)$ and E is a set of directed edges, $e_{ij} = (m_i, m_j), (1 \leq i \leq n, 1 \leq j \leq n, i \neq j)$ denoting the dependency relations among the modular tasks m_i and m_j [24, 66]. A module, m can be responsible for some independent computation tasks, such as any module, m_i can be responsible for some statistical analysis on a given dataset, while another module, m_j can be responsible for applying a machine learning model on the dataset and so on. The computation of any module, m_l is often further customized (e.g., number of *nodes*, *hidden layers* in case of a Neural Network Classifier and so on) by its corresponding configuration sets, C_l - a set of $\mathcal{P}_l$ available parameter configurations, $c_i \in C_l, (1 \leq i \leq \mathcal{P}_l)$. Hence, a collaborative SWfMS module can be generalized as Definition 5.1.

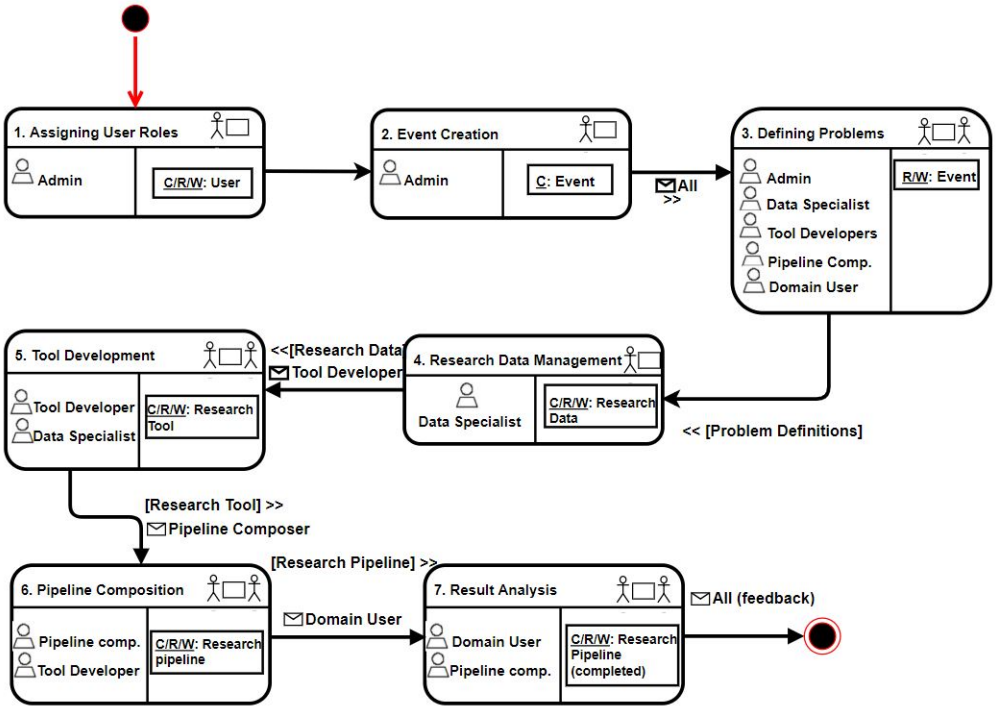

Fig. 2. Process Modeling for Collaborative Data Analysis Platform.

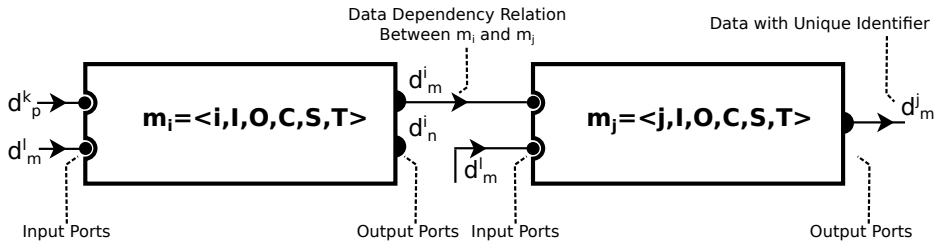

Fig. 3. Workflow DAG Formulation

*Definition 5.1 (**Collaborative Computational Module**).* A computational task module is generalized as a tuple, $m =< id, I, O, C, S, T >$ where id is the unique identifier of the module in a workflow, I and O are the corresponding set of supported input and output data formats respectively by the module, C is a set of $\mathcal{P}$ different parameter settings or configurations, S is the modular source code that executes based on I, O, C and finally, T is the set of possible states of the module (i.e., ready, running, success, failed, aborted) in an execution.

The generalized definition of the computational modules allows collaborators to plugin, reuse and share the set of tools with others for a given data analysis task.

5.1.2 Workflow Models and DAG Formulation. While a computational module from the toolbox is responsible for a given data analysis or manipulation task, a set of such modules are combined

together forming a workflow towards solving a more complex data analysis problem. Fig. 3 illustrates a dataflow relation of a workflow between two arbitrary computational modules- m_i and m_j. SciWorCS assigns unique identifier to each of the modules comprising a given workflow, such as $m_i =< id : i, I, O, C, S, T >$ and $m_j =< id : j, I, O, C, S, T >$. As a module from the toolbox can be used zero to multiple times, the identifiers are used to uniquely identify all the input-output ports in a given workflow. For example, we consider an arbitrary module- m from the toolbox, that contains two input ports- x, y, and one output port- z. We assume the module get a unique identifier- i on usage to a workflow. Hence, its input-output ports are encapsulated with the identifier, such as $m_i =< id : i, I : [x_i, y_i], O : [z_i], C, S, T >$, to uniquely identify in the workflow. The output dataset generated from an output port of a module is also labeled as the port identifier, which is used for forming the data dependency relation among the workflow modules. For example, as illustrated in Fig. 3, the module m_i is a predecessor of the module m_j where, one of its output datasets with reference- d_m^i from an output port is linked as the input dataset reference for an input port of module, m_j.

5.2 Consistency Management System for Collaborative DAG Formulation

One of the most important challenges for any real-time collaborative system is the consistency management of the shared objects in the face of concurrent conflicting operations by the collaborators [79, 80]. Sun and Chen [79] studied two primary important requirements for any collaborative system to be successful: i) *High Responsiveness:* Provided the non-deterministic communication latency, a collaborative system need to be light-weight with least amount of delay for a collaborator's action and its corresponding effect, and ii) *High Concurrency:* The collaborative system should be able to maintain a consistent state in spite of multiple concurrent update operations from the collaborators.

As illustrated in Fig. 1, our proposed architecture for collaborative SWfMS adapts a *hybrid style* for ensuring the high responsiveness in the data analysis process via direct client-client communication and also client-server communication for workflow components (i.e., data, module details and so on) and execution. That is, instead of running a single instance of the collaborative SWfMS for all the collaborators, we keep a local copy of the SWfMS for the each of the collaborators, where the corresponding update operation information are broadcast to the collaborators via simple light-weight message passing for synchronization. However, while our adaption of the replicated architecture facilitates the high responsiveness, it also comes with an added challenge of consistency management in the face of concurrent conflicting operations by the collaborators. For example, assuming the collaborative workflow DAG composition as illustrated in Fig. 3, two collaborators might concurrently update the incoming datalink d_m^l to two different values, x and y, where, $x \neq y$. A conflicting situation, hence occurs on message passing the update information to the other collaborator.

The consistency management in collaborative SWfMSs can often be different than that of the collaborative text or graphics editing systems for the complex dependency relations among the workflow modules [54, 73, 88]. The scientific workflows often being highly structured, any minor change of any attribute can significantly affect the execution behavior of the descendant sub-workflow [54, 74, 87]. A number of locking schemes have been proposed in recent years for facilitating the consistency management in dependency graphs [39, 75] and scientific workflow DAGs [54, 73, 74, 87, 88]. The locking schemes were proposed targeting the varying use-case scenarios such as, locking schemes [87] following *Robert's Rules of Order (RRO)* [50], locking schemes supporting *Backdoor Communication* [88], locking schemes targeting *granular controls* [20, 54, 74] and so on. While we propose a high-level architecture, the appropriate locking schemes can be selected and implemented as per the requirement of the specific collaborative SWfMSs.

5.3 Identification of Roles and Access Controls

Collaborative work involving different researchers or stakeholders face different challenges in its development phases. For example, creating a clear team vision or specifying the problems to solve by any particular research groups can often be confusing and thus reducing the overall team potential. This problem with role conflict, ambiguous problem statements to solve or overlapping of work by different researchers cost significant amounts of time and money [6]. This type of ambiguity and unclear definition of problem statements to the worse create stress and dissatisfaction among the research team slowing down the whole process [89]. So creating several distinguishable roles and assigning some specific tasks to each of them might show many possibilities in the improvement of productivity of the whole research team.

(i) *Domain User*: In workflow management system a large range of tools are accessed by domain users for scientific data analysis. Each tool is designed for a particular task. One tool's output is fed to the next tool for execution and so on. As domain users need to handle different tools, one might be expert with one settings and another might be others parameter setting. Scientific tools usually come up with lots of settings.
(ii) *Pipeline Composer*: This job will be done by an e-science developers who will make sure the system allows a wide range workflows for the domain users. While composing workflows, he may need to contact with a tool developers in case of any errors or missing of any tool.
(iii) *Tool developer*: Tool developers develop the scientific tools. He may need to collaborate with domain users for running the tools for testing the tool with real world settings.
(iv) *Data specialist*: Domain users and pipeline composers need to collaborate with data specialists for accessing data for tools.

After defining the possible user roles, we then identify the possible interaction and collaboration among the user. We use *Collaborative Interactive Application Methodology (CIAM)* [52], to design the process model for the proposed method. For process modeling, we chose to use CIAM, because of its support for collaborative system design. A successful process modeling for ensuring all the important interaction and collaboration among the users requires detail information about the user's responsibilities and hence, might vary for different use-cases or requirements. Fig. 2 illustrates the designed process model using CIAM for our proposed architecture of collaborative data analysis platform. The process modeling or the access controls on the workflow components are facilitated using the 'Participation Table' and 'Responsibilities Modeling' of CIAM. Participation table helps to get a higher level of abstraction about the individual and collaborative responsibilities, where Responsibilities Modeling are used for detailing the responsibilities on the basis of different user roles [52, 53]. After carefully following the steps of CIAM [52], we designed the process model as illustrated in Fig. 2. The model demonstrates the interactions among the identified user roles for the distinguished steps for collaborative data analysis. The process model also demonstrates the access controls on different workflow components along with the user permissions, such as *Create (C)*, *Read (R)* or *Write (W)*. For example, the Admin privileges are assigned to specific users with *C/R/W* permission. The arrows in the diagram also illustrates the link among the processes comprising the collaborative members.

5.4 Job Management and Execution for Collaborative Setup

Job manager is responsible for the scheduling and execution of the job (i.e., representing a modular task) from the workflow models. It manages the dependencies and execution order of the jobs to maintain a dataflow oriented execution plan - a job is ready and ordered for execution only if all of its required input datasets are available (i.e., input dataset or produced without errors from prior jobs). It also maintains a job queue for managing the multiple execution requests from the

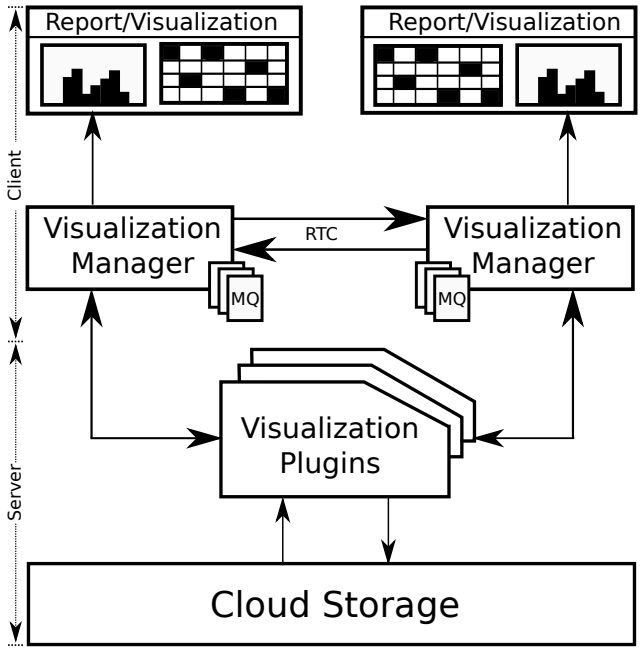

Fig. 4. Plugin-Based Collaborative Dataset Visualization

collaborators (such as, a sub-workflow of the entire data analysis steps for which a collaborator is responsible for and so on). The queued jobs are dispatched for local or cluster executions as per the configurations and implementation of the computational modular step, m. In case of cluster execution, the dispatched jobs are submitted to a cluster manager (such as, *Apache Spark* cluster manager [77]), which in turns are distributed across different worker nodes for parallel execution. The job manager also sends back the job execution status (e.g., *running*, *success* or *failed*) to the collaborators for real-time monitoring.

5.5 Plugin-Based Collaborative Dataset Visualization

Similar to the computational modules, a visualization plugin is generalized as a tuple, $v =< I, O, C, S >$, where I and O represent the corresponding set of input(s) and output(s), C is a set of different parameter settings or configurations and S is the modular source code for the corresponding data visualization. A plugin manager keep tracks of and pulls the available plugins for the usage. Finally, a plugin runner runs the plugins on the specified clone dataset for generating the visualization outputs. Fig. 4 illustrates the high-level architecture for the collaborative data analysis of the proposed method.

The visualization plugin manager also maintains a *Message Queue (MQ)* for the collaborative visualization information among the clients. The message structure might depend on the collaborative requirements. For example, in case of collaborative annotation, the message might contain the co-ordination information, color, shape types (e.g., rectangles or circles) and so on along with the corresponding user information. The visualization manager pulls the messages from the queue to render along with the dataset visualization for collaborative analysis. As the MQ is maintained directly among clients the responses are comparatively faster, keeping the collaborative dataset visualization more interactive.

5.6 User Awareness and Interaction

While the consistency management is one of the primary requirements of a collaborative system [78, 79], providing different methodologies for group communication towards problem-solving and decision making are also often very important towards the success of the system [17]. Using the message queue of our proposed architecture for client-client communication, the collaborative data analysis platform can adapt required implementation for user awareness and interaction from CSCW techniques. For example, using the user's mouse position information the 'Telepointer' features can be integrated to the system, text chat, collaborative white board and so on can also be implemented as per requirement of the collaborative system. We discuss implementation of some of such tools as proof of concept in Section 6.1.

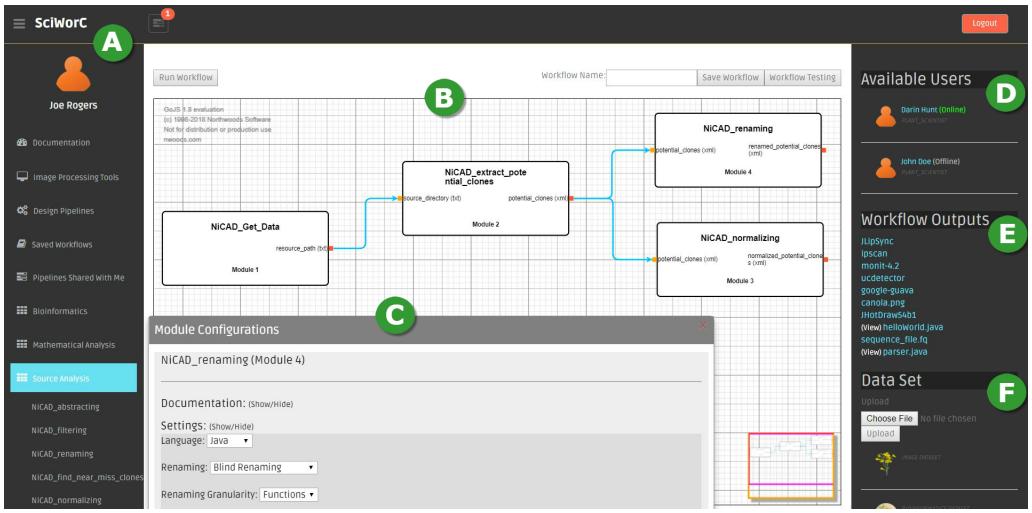

Fig. 5. User Interface Overview of SciWorCS.

6 EXPERIMENTS AND EVALUATION

In this section we present different experimental studies for evaluating the proposed architecture. In our conducted experiments we tried to address the following Research Questions (*RQ*).

- **RQ 1**: How can we design collaboration in scientific workflow management system?
- **RQ 2**: How can we implement a real-world collaborative SWfMS that is functional and performant by supporting consistency management, sub-workflow execution, visualization in various formats and so on?
- **RQ 3**: How can we utilize our collaborative platform for different scientific analysis domains?
- **RQ 4**: How does the proposed architecture impact the network overhead?
- **RQ 5**: How do scientists perceive real-time collaboration in data analysis?

We implemented *SciWorCS* as a proof of concept of the proposed architecture and also for answering RQ 1, which we discuss in Section 6.1. We then discuss different conducted experiments and real-world use-case scenarios in answering the above RQs 2, 3, 4 and 5 respectively.

6.1 SciWorCS as a Proof of Concept

We implemented the SciWorCS tool as a proof of concept of the proposed collaborative scientific workflow management system. The implementation is a cloud-based system. The SciWorCS is

implemented in Python programming language (e.g., *Python 2.7*). We used Python to leverage the larger number packages and library supports for different domains of data analysis. For example, a great numbers of bioinformatics [1, 84], image processing, machine learning [64] and so on tools and libraries are recently implemented in Python for its trending popularity. Besides, the tools those are written in a different programming language can also be added to SciWorCS toolbox using Python wrapper for the command line access.

We used *HTML5*, *CSS* and *JavaScript* for the client side programming and creating user interface for easier accessibility of SciWorCS core. SciWorCS provides an intuitive graphical user interface for defining the workflow DAG, -where the selected tools from the toolbox can be graphically connected, reorganized, zoomed in/out, modified and so on. The interactive and intuitive interface allows increased accessibility of SciWorCS. We used *GoJS* [26] - a JavaScript library- for implementing interactive diagrams in HTML. We used JavaScript Ajax for asynchronous server communications for obtaining different information from the server, such as availability of a dataset in the server, job status for a workflow execution and so on.

Fig. 5 demonstrates an overview of the SciWorCS user interface. The panel labeled as- 'A', contains all the workflow components such as, multiple toolboxes categorized as per the general data analysis tasks of the modular tools, saved workflows and shared workflows with collaborators. The composition of the workflow is done on panel 'B'. The selected modular tools from the toolbox appear in this panel where they can be connected defining the dataflow relation among them. As illustrated in the figure, the workflow data flow relation is presented in intuitive DAG representation. The modules can be configured using the corresponding attributes in the popup panel 'C', -which appears on *Mouse Left Double-Click* on any module from the workflow DAG. Panel 'D' shows a list of collaborators and their current online/offline status. The list of the workflow outputs are presented in panel 'E' and the new dataset can be browsed and uploaded to the server for analysis from the panel 'F'.

We also incorporate different communication tools for the group discussion and decision making in the process of collaborative data analysis. The textual communication tools include peer-to-peer and group chatting system. In addition to the textual communication tools, SciWorCS provide real-time Audio and Video streaming based communication system among the collaborators. As also illustrated in the figure, a *Collaborative Virtual Whiteboard (CVW)* also has been implemented in SciWorCS for aiding the group discussion and problem-solving. The CVW contains different simple tools (such as color selector, paintbrush size selector and so on) to manage the discussion process among the collaborators. In real-time groupware systems, *telepointers* (e.g., multiple cursors) are widely used to increase the group awareness [30]. Studies show that, in the context of CSCW, through the simple movement of telepointers, collaborators often communicate their focus of attention, gesture over the shared views and so on [29, 30]. For example, in terms of collaborative SWfMSs, telepointers often might be used by the collaborators to create group awareness about the individuals' focus on attention on sub-workflows, tools and so on. Hence, we also implement real-time telepointer communication systems among the collaborators in SciWorCS.

SciWorCS leverages *WebRTC* for the implementation of the communication tools. WebRTC provides real-time peer-to-peer communication along with audio-video streaming from modern browsers without any further requirements of software packages or tools [44]. Note that, while we have implemented different communication systems for the collaboration, their selection or usage patterns among collaborators might often be impacted by several factors, such as the collaborating group itself, the nature of data analysis tasks for collaboration and so on.

Answering RQ 1. The implementation of SciWorCS helps answering the RQ1 and demonstrates the potential and possibility of implementation of the proposed architecture. The interpreter based Python programming language was used for real-time plugin architecture, job management and

execution. JavaScript was used for making data-intensive Asynchronous server request and on the other hand, the WebRTC technology was used for light-weight Client to Client message passing for real-time collaboration.

6.2 Effects of Varying Locking Schemes

The lock on a module with higher dependency degree, ϕ (i.e., the total number of distinct descendant modules), can largely impact the overall collaboration scope of the workflow. We thus conducted several experimental studies with varying workflow tree structures to test the hypothesis. We considered six different dependency relations of the workflow trees and two different locking schemes - *Strict Module Locking* [20, 88] and *Attribute Level Locking Scheme* [54] as presented in Fig. 6.

The 2, 3 or 4 *regular workflow trees* in the figure represents three different structures, where every non-leaf workflow module has exactly 2, 3 or 4 child modules respectively. The 2, 3 or 4 *all connected workflow trees* on the other hand, represent some-what similar structures, but with higher dependency degree considerations where, any workflow module of level l, is dependent on all of the modules of level $(l-1)$ by direct incoming dataflow relation among them (i.e., except for the root workflow module).

As the methodology for simulating *short-read, long-thinking* pattern [87, 88] in the simulation experiments, we considered random *thinking time* [88] interval ranging from 10 ms to 15 ms in between any basic workflow operations i.e., i) Adding a new module to the workflow, ii) Adding a new datalink to/from a module, iii) Updating an attribute of a module and iv) Updating source/destination of a datalink.) execution by a collaborator. If the next thinking time is relatively longer (e.g., considered, >10 ms), the corresponding simulated collaborator releases any accessed object, making it available for other collaborators of the group. As for facilitating the consistency management, the locking schemes adapt some algorithms for managing the workflow component access to the collaborators. For evaluating the behavior in terms of collaborative SWfMSs, we considered different locking schemes to simulate the collaborative workflow composition. In the context of collaborative SWfMSs, the behavior of the locking schemes were evaluated in terms of *average waiting time*, - the average time required in between the access requests and grants of different workflow components by collaborators for the workflow composition.

The experimental results as illustrated in the figure, show a significant increase in average waiting time with the increase of overall dependency relation in case of strict module locking scheme. For example, in case of strict locking, for 2 all connected workflow tree the average waiting time raises up to around 1520 ms in comparison to proposed attribute level locking scheme, which is approximately 958 ms. That is, the average ratio of reducing the overall waiting time by the proposed method is around 0.63 (i.e., $958 * 100/1520 = 63\%$) in case of 2 all connected workflow tree structures. Similarly in case of 2 regular workflow tree structures, the average waiting time with 29 collaborators raises up to around 1243 ms and 726 ms for strict module locking and proposed locking schemes respectively. It is also noticeable from the graphs that, the difference in average waiting time between the locking schemes increases significantly with the increase of collaborating group size. For example, both the locking schemes show more or less similar average waiting time when the group contains five or less number of collaborators, however the average waiting time increases noticeably with the increase of group size for strict locking scheme in comparison to the proposed locking scheme.

Answering RQ 2. The experimental study demonstrates that the proposed architecture can adapt different variance of locking schemes. From the graph, the study reveals that the proposed architecture shows the scaling capacity in terms of increasing number of collaborators. While the average waiting time increases with the increased number of collaborators for the access of limited

workflow components, its comparative lower increase demonstrates the scaling capacity of the proposed architecture. Besides, the behavior can further be improved with the adaption of more efficient locking scheme.

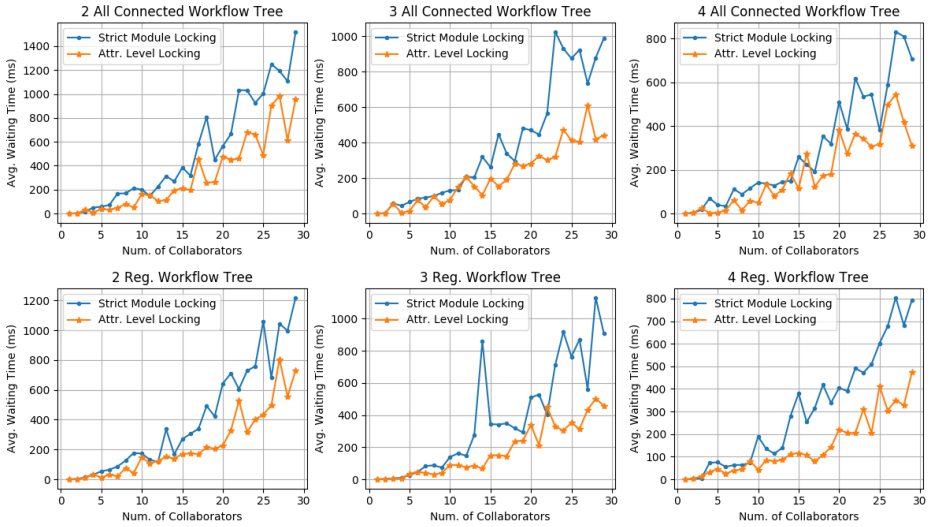

Fig. 6. Effects of Locking Schemes on Performance.

6.3 Use-Case Evaluation for Collaborative Data Analysis

Existing studies show several motivations and use-cases towards collaborative SWfMSs [46, 46, 87, 88]. Different experimental studies can exploit the added advantages of collaboration for accelerating the overall process [88].

6.3.1 Code Clone Detection (Software Repository Analysis). *Code clones* are similar pairs of code fragments in software systems [55, 70, 71]. Studies show that, on average software systems often contain 7% to 23% of codes that are copied from one location to another (e.g., code clones) [2, 38, 71]. Code clones are often responsible for inflating the overall software maintenance cost, and hence their successful detection has been one of the research problem in the domain of software engineering [71, 81].

Throughout the life-cycle of a software system, the code clone fragments often undergo several changes, such as identifier name changes, addition/edition/removal of several statements, changes in the presentation and so on [70, 71]. Code clone detection techniques hence, undergo several pre-processing and transformation steps, such as *pretty-printing* [70, 72], *normalization of the identifiers* [37, 72], *forming syntax tree* [41] of the code fragments and so on, prior to applying any matching algorithms which can be aided with SWfMSs. The different computational units can be plugged-in as re-usable module as proposed in the architecture for scientific data analysis.

For the use-case, we leverage the modular steps from *NICAD* [72] — a widely used code clone detection tool, to demonstrate the re-usability of the workflow modules towards clone detection process as per the given requirements. As input, the workflow takes the target software system

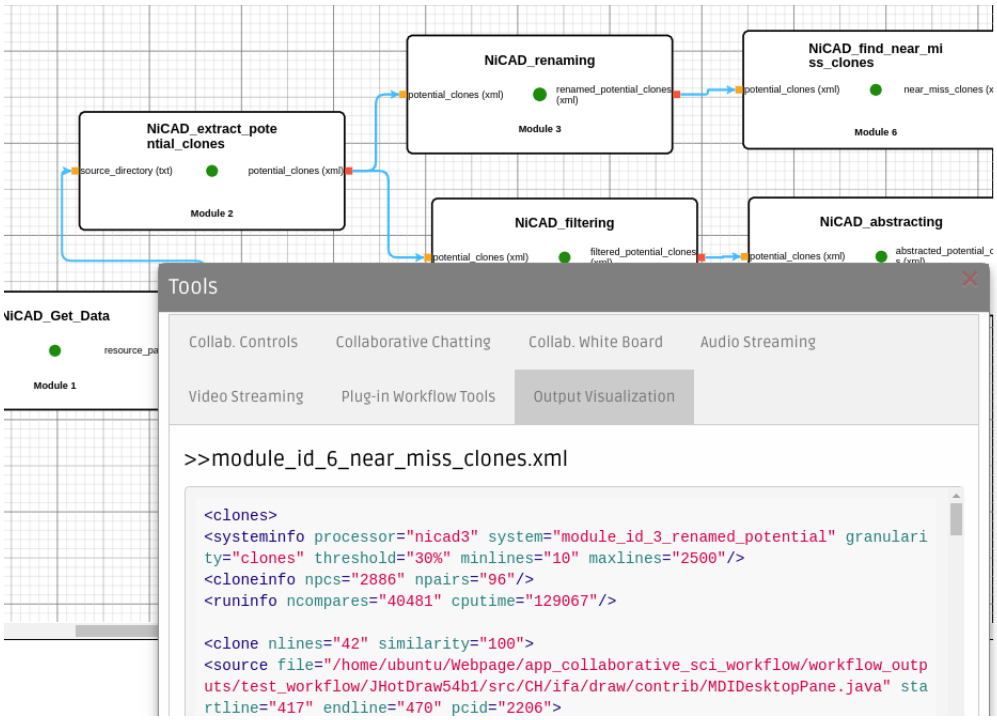

Fig. 7. Code Clone Detection Workflow in SciWorCS.

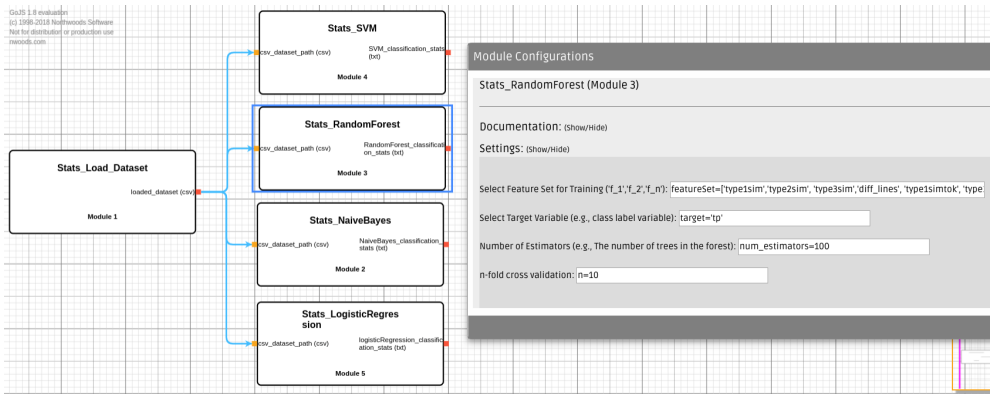

Fig. 8. Applying Machine Learning in SciWorCS for Clone Validation.

repository and outputs the detected code clone pairs along with different statistics of the detected clones (e.g., number of detected clones, pair-wise similarity values and so on). Fig. 7 illustrates the scientific workflow comprising of the plugged-in modules to SciWorCS for code clone detection. On execution of the workflow on supplied data (i.e., software source codes), it detects and outputs the code clone pairs available in the source codes. The figure also demonstrates the visualization of the output dataset as proposed in our architecture.

Collaboration Scope. The definition of code clone given a pair of code fragments is often subjective

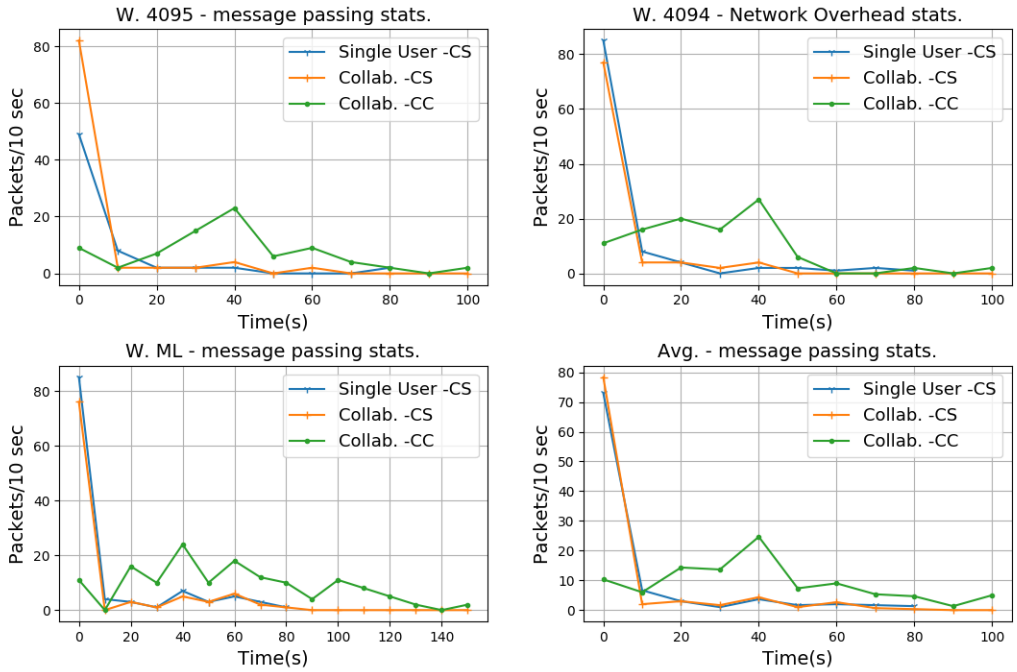

Fig. 9. Experimental results on message passing behavior of proposed collaborative SWfMSs architecture.

Table 2. Classification Accuracy Comparison for different Machine Learning Models.

#	Machine Learning Classifier	Classific. Acc. (%)
1	Random Forest	84.47
2	Random Tree	79.84
3	Naive Bayes	83
4	Bayes Network	81.79
5	Naive Bayes Updateable	82.99
6	Logistic Regression	85.06

[70, 71]. For example, in a related study, Yang et al. [86] conducted a survey, where users were provided the same clone sets (e.g., detected by clone detection tools) for validation. The study reported a significant variations among the users in validating the same clone sets (e.g., for the same provided clone sets, the number of decided true positive code clones varied within a range of 4.76% to 23.81% for different users). Hence, SciWorCS can be leveraged for machine learning based automatic clone classification (i.e., True/False Clones). For example, Fig. 8 demonstrates the usage of different machine learning algorithms towards automatic code clone validation. Initially, collaborators can work on building the manually validated dataset, which are then used by the plugged-in machine models in SciWorCS. For example, Table 2 demonstrates the obtained classification accuracy for some of the plugged-in machine learning models in SciWorCS for code clone classification.

Besides, finding suitable configuration for a given scenario of clone detection is also a research problem in the software research domain [85]. Hence, the proposed architecture exhibits significant potential in allowing the researchers towards collaboratively configuration searching and converging to the subjective preferences of detected clones.

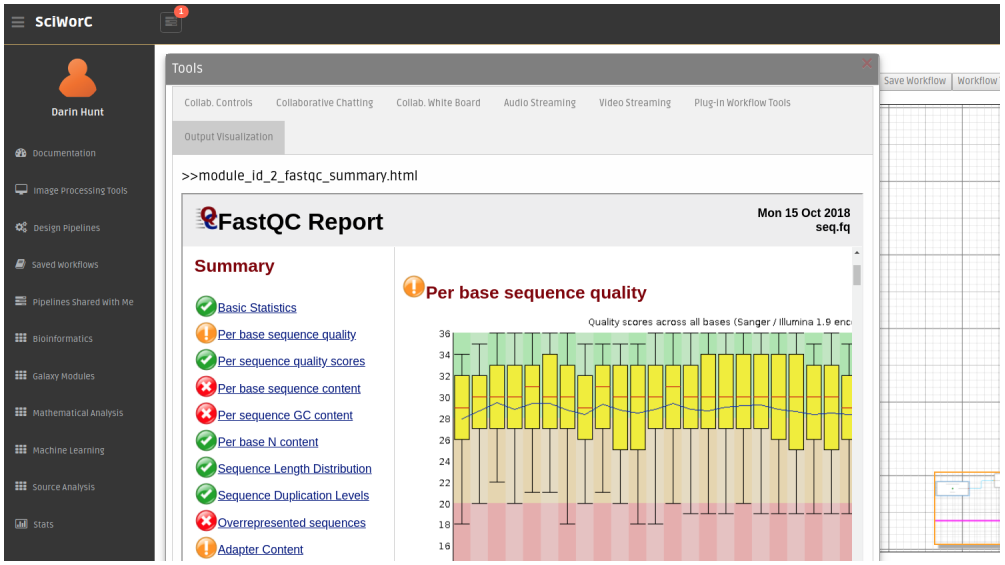

Fig. 10. FastQC quality measures on example sequence file.

6.3.2 QC Report of FastQ File with FastQC (Bioinformatics). Different *Quality Control (QC)* tools are widely used in the field of Bioinformatics to investigate any potential problem or check the quality of the input sequence files [7]. Here, in this use-case we demonstrate the usage of QC workflow for analysis of sequence files.

FastQ is a popular sequence file containing *Nucleotide* sequence data with associated quality scores [7, 18]. *FastQC* is a widely-used tool for QC report generation and analysis of FastQ sequence files [18]. In addition to the FastQ sequence files, the tool also accepts inputs as *Sequence Alignment Map* (SAM) or *Binary Alignment Map* (BAM) formats—runs a series of tests and generates corresponding QC reports [18]. FastQC also generates tables, graphs and HTML based permanent report for overall visualization of the QC reports.

Fig. 10 illustrates a sample workflow illustrating the visualization of the generated report from FastQC. Note that, as FastQC, other visualization tools can be plugged-in to SciWorCS—where the generated outputs are used by the framework for visualization.

Collaboration Scope. The quality analysis tools are important to ensure that there are no hidden problems on the sequence files—which might be difficult to detect and recover from- at some later stages of the analysis of the sequence file [18]. Our proposed architecture provides the collaborative visualization support, which can be used by the researchers for the analysis and reviewing of the sequence file quality.

Answering RQ 3. The experimental study reveals the significant potential and collaboration scope of the proposed architecture in terms of real-world data analysis steps. While the two above scenarios demonstrate primary use-case, the proposed architecture can be extended as per the requirements of the specific data analysis problems.

Table 3. Considered Arbitrary Scientific Workflows from *myExperiments* [14] for the Study.

W. ID	Type	Workflow Summary
4095 [58]	Bio.	Paired-end reads assembly after FastQ groomer using a Migale modified version of Velvet tool.
4094 [59]	Bio.	Workflow used when applying the CPB2012 Basic Protocol 3; Peaks for ChIP-seq data using MACS14.
[36]	ML	Open source *The Titanic* dataset classification.

6.4 Experiments on Message Passing Behavior

Our proposed architecture uses simple message passing for the orchestration of used computational modules and also for configuration of the overall workflow DAG and module settings in a replicated architecture. The message passing in case of collaborative data analysis platform can be different than that of graphics or text based collaborative systems, where the message passing among the clients are the corresponding typed characters [79, 80]. In our proposed architecture, while the communication varies as per the computational modules (i.e., implementation structure, lengths and so on) or module configurations, the message passing information among the clients remains somewhat fixed for the module or DAG updates.

We were interested to testing the message passing behavior and network overhead of our proposed architecture. For the experiment, we used two real-world bioinformatics modules from *myExperiment* and one machine learning based classification workflow for the data analysis tasks as listed in Table 3. The workflow compositions were simulated both for single-user and collaborative setups. Since our analysis is on message passing behavior between two clients and server-client, we considered two independent simulator instance for collaborative setups. We repeated the experiments three times to mitigate possible biases. We used *Wireshark*[63] - a network analyzer tool, to track the message passing behavior.

Fig. 9 illustrates the obtained results from the experiments for the above three different data analysis workflows and also their averaged results. All the graph demonstrates relative higher amount of packets transfer (i.e., around *80 Packets/10 sec*) for both *Client-Server* and *Client-Client* for network and connection initialization. The message passing for *Client-Server* is more or less the same in both of the single and collaborative use-cases for requesting and pulling out the required modules from the server. The behavior is clearly noticeable from the average plots. It is also noticeable from the plots that the implementation of the specific modules are correlated with the overall *Client-Server* message passing. For example, Bioinformatics workflows (i.e., *W. 4095* and *W. 4094*), the module implementations are as wrapper (i.e., higher level abstraction) to the original source code. The wrapper structure are generally minimal in nature in oppose to complete original source codes, and hence resulted in lesser *Client-Server* message passing, such as *1 to 4 Packets/10 sec* for the use cases. On the other hand, the Machine Learning (ML) workflow modules contained original source implementation and hence, the message passing shows relatively higher rate, such as in the range of *1 to 7 Packets/10 sec* for the use-case.

For any module addition/deletion, the *Client-Client* message passing is somewhat similar (i.e., comprising the unique module identifier information) irrespective of the specific module. However, the *Client-Client* message passing is also dependent on the module configurations and datalink relation updates among them. For example, as noticeable from the average plot, the *Client-Client*

message passing ranges between 0 to 10 Packets / 10 sec after 50 sec of the workflow composition for corresponding workflow update operations (i.e., module configurations and datalink relations), while there are minimal *Client-Server* message passing.

Answering RQ 4. The study results exhibit minimal network for the proposed architecture. While the Client-Client message passing depends on the specific workflow structure and configurations, the *Client-Server* message passing shows fairly similar behavior when compared with single user based data analysis, which is promising for scaling the proposed architecture for increased number of collaborators.

6.5 User Study

In order to investigate how research scientists perceive collaboration data analysis (RQ5), we conducted a user study with the proposed collaborative SWfMS. This user study helped us get insights how users make use of the system, e.g. what styles of works collaborators engage in for scientific workflow composition for collaborative data analysis. We used four well known real world bioinformatics data analysis tasks or workflows from myExperiment [14] as described in Table 4.

Table 4. Considered well-known Scientific Workflows from *myExperiments* [14] for the User Study.

W. ID	Workflow Summary
4095 [58]	Paired-end reads assembly after FastQ groomer using a Migale modified version of Velvet tool.
4094 [59]	Workflow used when applying the CPB2012 Basic Protocol 3; Peaks for ChIP-seq data using MACS14.
2944 [56]	Transform FastQ to FASTA, using the tools, Groomer, Filter FastQ, FastQ Trimmer.
2939 [57]	Retrieves Genome and SNP data from UCSC for a particular chromosome. Finds Exons from SNPs.

In total these workflows require 19 unique computational modules, which were integrated to our collaborative framework for the study. The computational modules are the building blocks of the workflows for collaborative composition of the study. To mitigate the bias of problem solving complexity while acquiring the collaborative composition patterns, the workflow structures were printed and provided to the participants for collaborative composition.

For a given workflow, participants had to select (i.e., via Mouse Left-Click) the required computational modules available in the tool panel (i.e., Panel 'A', Figure 5) of the collaborative framework. The selected modules appear in the composition panel (i.e., Panel 'B', Figure 5) for devising the required datalink relations among them via *Mouse Left-Click and Dragging* among the corresponding input/output ports of the modules. Participants follow a series of collaborative revisions and updates on a workflow such as, module configuration changes (i.e., Panel 'C', Figure 5, on *Double-Clicking* a module), *delete/update/addition* operations on the workflow components (i.e., modules or datalinks) and so on to complete the composition of the target workflow. Note that for consistency management we used attribute level locking scheme as it reduces average waiting time for the participant as per the simulated study described by Mostaeen et al. [54]. This locking scheme allows collaborators to work independently on selected region of the workflow (i.e., sub-workflow).

While the selected locking scheme ensures consistent composition in the face of conflicting operations, participants use different available user interaction tools of the framework for orchestrating the collaborative composition of the workflow. The telepointer (i.e., multiple cursors) information is passed among the collaborators for real time group awareness on their location, movement and probable focus of attention [29] in the collaborative workflow. Collaborators also invoke other communication tools (i.e., audio communication, video conferencing, textual group/P2P chatting) of the framework for discussion and convergence on a plan. In addition to that the collaborative discussion process is also assisted by the framework's data visualization and virtual white board tools.

Ten graduate students from a local university participated in the experiments. The participants were split in five groups, each comprising of two members for collaborative workflow composition. The participants were introduced and trialed with SciWorCS editor prior to starting the user study.

6.5.1 Results of our User Study. All the collaborating group could successfully complete the given tasks for the study with an average time of 15 minutes. For the collaborative composition task in this study, we found that collaborators more often adapt the divide and conquer work approach towards completing the composition. The clear target for the mere composition task in collaboration, resulted in more or less fair task splits among the collaborators. Prior to starting the task, the collaborators were found to make plan via discussion for approaching the problem. In addition to the discussion, individual collaborator were also found to directly contribute to the composition (e.g., via module/datalink addition, deletion and so on) - as opposed to the 'scribe' style of work (e.g., where a single collaborator is responsible for entire composition). For example, for a collaborative composition two collaborators had - 65% and 35% splits of operations (e.g., module/datalink addition, deletion and so on) out of total 105 operations for the workflow composition; combined for the workflows presented in Table 4. Besides, the total engagement in the discussion also showed a fair split of 55% and 45%. Similar trends were also found among other collaborating groups for the composition task.

From an individual perspective in a collaborative group, a participant's results also exhibit contribution in different aspects of the composition. For example, in a collaborative group, a member contributed 15% in discussion, 50% in edit operations (e.g., module/datalink addition, deletion and so on) and 35% in other management operations towards the composition, such as DAG view update, access request/wait/release and so on, out of all the generated events by the collaborator in the composition. Similarly, another collaborator also found to split the self events 23% in decision making discussion, 47% in edit operations and 30% in other management operations. That is, the results demonstrate that, the collaborators in overall, adapted the task splits and also individuals contributed in different aspects of the composition. For example, Fig. 11 illustrates the engagement of the participants in collaborative composition. For more granular pattern visualization, a whole collaborative composition (e.g., from log) has been divided into two sessions and plotted the contribution of the collaborators for the corresponding sessions. The graph illustrates that, the collaborators showed their engagement in comprehending the composition problem across different sessions.

The chat log of the collaborators also exhibits the similar patterns, such as: - *'Collab. #1: ... Do you have a way to proceed? I was thinking lets get [add to workflow] the input modules first?...', 'Collab. #2: ... Alright. Go ahead ...', 'Collab. #1: ... I have set all the user inputs [modules]... Would you like to do [add to the workflow] the second layer [from the provided reference]?...'* .

Answering RQ5. The preliminary user study shows that users utilizes collaboration facilities of SciWorCS in scientific data analysis. This kind of system can be helpful for complex data analysis tasks and can increase the overall productivity. For understanding users behavior more

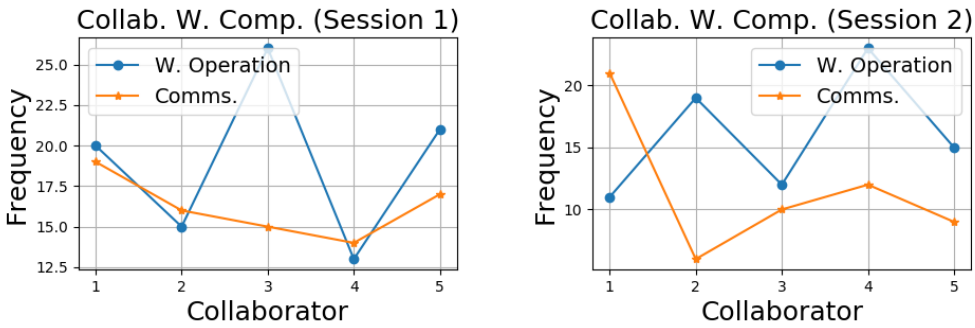

Fig. 11. Collaborative Composition Work Patterns (where, *W. Operation*: workflow update operations such as, module addition/remove/update or datalink addition/remove/update. *Comms.*: communication among collaborators. *Frequency*: frequency of *W. Operation* or *Comms.* in collaborative setups.)

appropriately, in future we will involve more users and we will use NASA TLX questionnaire to identify their behavior by measuring cognitive workload, performance, effort and frustration.

7 RESULT DISCUSSION

We evaluated our proposed architecture for a collaborative data analysis platform in different setups in Section 6. Our implementation of SciWorCS as a proof of concept of the proposed architecture demonstrates that it can address the primary requirements of a collaborative data analysis platform, including plugin support, consistency management for collaborative workflow composition, collaborative job planning and execution, user interaction tools for problem solving, and collaborative modelling for a variety of user roles.

We also evaluated the different components of the proposed architecture using SciWorCS. Our study demonstrates that the proposed architecture can adapt to various locking scheme implementations, as per the requirements and use case scenarios. As well, the simulated studies for varying scientific data analysis workflows (i.e., 2, 3 and 4 all-connected trees and 2, 3 and 4 regular workflow trees) and for 1 to 30 collaborators, demonstrates the proposed architecture's scaling capability and also its performance in terms of different locking schemes.

We also experimented to determine the network overhead of the proposed architecture for collaborative workflow composition. The experiments demonstrate that the proposed method for collaborative composition results in minimal overhead in contrast to single user based composition. We used real-world data analysis workflows for Bioinformatics and Software Repository Analysis along with their scope of collaboration, where the proposed architecture exhibits significant potential.

Our small scale user study demonstrates that SciWorCS in the domain of bioinformatics has the potential to support users to do real-world bioinformatic tasks. Although our user study does not describe user behaviour in multiple dimensions (such as ease of use, cognitive workload, and degree of frustration) and for different domains, it shows that users can successfully complete their assigned tasks within a reasonable amount of time (on average 15 minutes) while being involved with various collaborative activities (such as shared editing, chatting, and shared visualization).

8 THREATS TO THE VALIDITY

We evaluated the proposed architecture using computer generated simulations in terms of different setups and metrics, such as the effect of varying locking schemes, impacts of different workflow structures, network overhead and so on. While the simulated studies allowed us to test the scalability of the proposed methods, it also exhibit threats for any result biases which is a common threat for any simulation studies. However, since our experiments we primarily focused on the scalability and performance of the proposed architecture in extreme cases, we chose to conduct simulated studies with varying setups and scales as opposed to experiments with a few users, so that we can map the results in the case of limited number of users. In order to further mitigate bias, the experiments were conducted on the same machine and also repeated a number of times to use their average values for convergence. For the user study, we mostly involved graduate students as the study participants and thus their experience with our proposed platform may not reflect what a domain expert would experience. However, in order to at least partially mitigate the issue, we chose the participants who are knowledgeable of the applications of scientific workflows in the bioinformatics domain.

9 CONCLUSION AND FUTURE WORK

Realizing the compelling need for collaborative data analysis, several methods or techniques have been proposed and developed in recent years [73–76, 87, 88]. In this paper, we identified the primary challenges for a collaborative data analysis platform in contrast to collaborative text or graphics editing systems. From these findings, we then proposed an architecture for a collaborative data analysis platform. The architecture addresses plugin-based tool integration, role-based access control for workflow components, independent sub-workflow execution and monitoring for supporting sub-group collaborations, integrated communication technologies for group discussions, decision making or problem solving, and plugin-based visualizations. We leveraged Scientific Workflow Management System (SWfMS) functionality for handling the underlying data-intensive processing in our proposed collaborative data analysis architecture. As a proof of concept, we also developed a collaborative SWfMS — SciWorCS. We presented different technical features of SciWorCS and also presented several real-world use-cases for scientific workflow composition, execution, and data visualization. Our proposed architecture demonstrates promising results when evaluated in terms of different real-world scenarios of scientific workflow collaboration. Our preliminary user study in the bioinformatics domain also shows the promise of the platform.

In the future we would like to use SciWorCS to investigate users' behavior in different domains, such as Plant Phenotyping and Genotyping, Hydrological Modelling, and Source Code Analysis. We will utilize the NASA-TLX [31] questionnaire to analyze users' behaviour in these domains. In particular, we want to compare the performance of different locking schemes (discussed in Section 5) for different scientific analysis which will demand both short-term and long-term thinking behaviour from users. By analyzing the experimental data we want to be able to recommend which locking scheme is suitable for which situations.

REFERENCES

[1] Enis Afgan, Dannon Baker, Marius Van den Beek, Daniel Blankenberg, Dave Bouvier, Martin Čech, John Chilton, Dave Clements, Nate Coraor, Carl Eberhard, et al. 2016. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic acids research* 44, W1 (2016), W3–W10.

[2] Brenda S Baker. 1995. On finding duplication and near-duplication in large software systems. In *Reverse Engineering, 1995., Proceedings of 2nd Working Conference on*. IEEE, 86–95.

[3] Adam Barker and Jano Van Hemert. 2007. Scientific workflow: a survey and research directions. In *International Conference on Parallel Processing and Applied Mathematics*. Springer, 746–753.

[4] Aaron Bauer and Zoran Popovic. 2017. Collaborative Problem Solving in an Open-Ended Scientific Discovery Game. *PACMHCI* 1, CSCW (2017), 22–1.

[5] Fahima Bhuyan, Shiyong Lu, Robert Reynolds, Ishtiaq Ahmed, and Jia Zhang. 2018. Quality Analysis for Scientific Workflow Provenance Access Control Policies. In *2018 IEEE International Conference on Services Computing (SCC)*. IEEE, 261–264.

[6] Robert P Bostrom. 1980. Role conflict and ambiguity: Critical variables in the MIS user-designer relationship. In *Proceedings of the seventeenth annual computer personnel research conference*. ACM, 88–115.

[7] Joseph Brown, Meg Pirrung, and Lee Ann McCue. 2017. FQC Dashboard: integrates FastQC results into a web-based, interactive, and extensible FASTQ quality control tool. *Bioinformatics* 33, 19 (2017), 3137–3139.

[8] Jeffrey L Brown, Clayton S Ferner, Thomas C Hudson, Ann E Stapleton, Ronald J Vetter, Tristan Carland, Andrew Martin, Jerry Martin, Allen Rawls, William J Shipman, et al. 2005. Gridnexus: A grid services scientific workflow system. *International Journal of Computer Information Science (IJCIS)* 6, 2 (2005), 72–82.

[9] Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. 2006. VisTrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. ACM, 745–747.

[10] Esther Care, Patrick Griffin, Claire Scoular, Nafisa Awwal, and Nathan Zoanetti. 2015. Collaborative problem solving tasks. In *Assessment and teaching of 21st century skills*. Springer, 85–104.

[11] Artem Chebotko, Shiyong Lu, Seunghan Chang, Farshad Fotouhi, and Ping Yang. 2010. Secure abstraction views for scientific workflow provenance querying. *IEEE Transactions on Services Computing* 4 (2010), 322–337.

[12] Yuan Cheng, Fazhi He, Yiqi Wu, and Dejun Zhang. 2016. Meta-operation conflict resolution for human–human interaction in collaborative feature-based CAD systems. *Cluster Computing* 19, 1 (2016), 237–253.

[13] Brian Corrie and Todd Zimmerman. 2009. Build It: Will They Come? In *Media Space 20+ Years of Mediated Life*. Springer, 393–413.

[14] David De, Roure Carole, and Goble Robert Stevens. 2008. The design and realisation of the myexperiment virtual research environment for social sharing of workflows. (2008).

[15] Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. 2009. Workflows and e-Science: An overview of workflow system features and capabilities. *Future generation computer systems* 25, 5 (2009), 528–540.

[16] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, et al. 2005. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming* 13, 3 (2005), 219–237.

[17] Joanna DeFranco-Tommarello and F Deek. 2002. Collaborative software development: a discussion of problem solving models and groupware technologies. In *hicss*. IEEE, 41.

[18] FastQC. [n. d.]. A quality control tool for high throughput sequence data. http://www.bioinformatics.babraham.ac.uk/projects/fastqc/.

[19] Xubo Fei and Shiyong Lu. 2012. A dataflow-based scientific workflow composition framework. *IEEE Transactions on Services Computing* 5, 1 (2012), 45–58.

[20] Xubo Fei, Shiyong Lu, and Jia Zhang. 2011. A Granular Concurrency Control for Collaborative Scientific Workflow Composition. In *Services Computing (SCC), 2011 IEEE International Conference on*. IEEE, 410–417.

[21] Stephen M Fiore and Travis J Wiltshire. 2016. Technology as teammate: Examining the role of external cognition in support of team cognitive processes. *Frontiers in psychology* 7 (2016), 1531.

[22] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T Silva. 2008. Provenance for computational tasks: A survey. *Computing in Science & Engineering* 10, 3 (2008).

[23] Liping Gao, Fangyu Yu, Qingkui Chen, and Naixue Xiong. 2016. Consistency maintenance of Do and Undo/Redo operations in real-time collaborative bitmap editing systems. *Cluster Computing* 19, 1 (2016), 255–267.

[24] Ritu Garg and Awadhesh Kumar Singh. 2015. Adaptive workflow scheduling in grid computing based on dynamic resource availability. *Engineering Science and Technology, an International Journal* 18, 2 (2015), 256–269.

[25] Jeremy Goecks, Anton Nekrutenko, and James Taylor. 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology* 11, 8 (2010), R86.

[26] GoJS. [n. d.]. Interactive JavaScript Diagrams in HTML. https://gojs.net/latest/index.html.

[27] Ian Goldin. 2010. *World wide research: Reshaping the sciences and humanities*. MIT Press.

[28] Katharina Görlach, Mirko Sonntag, Dimka Karastoyanova, Frank Leymann, and Michael Reiter. 2011. Conventional workflow technology for scientific simulation. In *Guide to e-Science*. Springer, 323–352.

[29] Saul Greenberg, Carl Gutwin, and Mark Roseman. 1996. Semantic telepointers for groupware. In *Computer-Human Interaction, 1996. Proceedings., Sixth Australian Conference on*. IEEE, 54–61.

[30] Carl Gutwin and Saul Greenberg. 1995. Support for group awareness in real-time desktop conferences. (1995).

[31] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.

[32] Mark Hartswood, Rob Procter, Mark Rouncefield, and Roger Slack. 2003. Making a case in medical work: implications for the electronic medical record. *Computer Supported Cooperative Work (CSCW)* 12, 3 (2003), 241–266.
[33] David Hollingsworth and UK Hampshire. 1995. Workflow management coalition: The workflow reference model. *Document Number TC00-1003* 19 (1995), 16.
[34] Marina Jirotka, Charlotte P Lee, and Gary M Olson. 2013. Supporting scientific collaboration: Methods, tools and concepts. *Computer Supported Cooperative Work (CSCW)* 22, 4-6 (2013), 667–715.
[35] Marina Jirotka, Rob Procter, Mark Hartswood, Roger Slack, Andrew Simpson, Catelijne Coopmans, Chris Hinds, and Alex Voss. 2005. Collaboration and trust in healthcare innovation: The eDiaMoND case study. *Computer Supported Cooperative Work (CSCW)* 14, 4 (2005), 369–398.
[36] Kaggle. [n. d.]. Titanic: Machine Learning from Disaster. https://www.kaggle.com/c/titanic/data.
[37] Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue. 2002. CCFinder: a multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions on Software Engineering* 28, 7 (2002), 654–670.
[38] Cory J Kapser and Michael W Godfrey. 2006. Supporting the analysis of clones in software systems. *Journal of Software: Evolution and Process* 18, 2 (2006), 61–82.
[39] VR Kavitha and N Suresh Kumar. 2013. A Method for identifying loops in a Workflow using Petri Nets. *Life Science Journal* 10, 3 (2013).
[40] Terhi Kilamo, Antti Nieminen, Janne Lautamäki, Timo Aho, Johannes Koskinen, Jarmo Palviainen, and Tommi Mikkonen. 2014. Knowledge transfer in collaborative teams: experiences from a two-week code camp. In *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 264–271.
[41] Rainer Koschke, Raimar Falke, and Pierre Frenzel. 2006. Clone detection using abstract syntax suffix trees. In *Reverse Engineering, 2006. WCRE'06. 13th Working Conference on*. IEEE, 253–262.
[42] Cui Lin, Shiyong Lu, Xubo Fei, Artem Chebotko, Darshan Pai, Zhaoqiang Lai, Farshad Fotouhi, and Jing Hua. 2009. A reference architecture for scientific workflow management systems and the VIEW SOA solution. *IEEE Transactions on Services Computing* 2, 1 (2009), 79–92.
[43] Ji Liu, Esther Pacitti, Patrick Valduriez, and Marta Mattoso. 2015. A survey of data-intensive scientific workflow management. *Journal of Grid Computing* 13, 4 (2015), 457–493.
[44] Salvatore Loreto and Simon Pietro Romano. 2014. *Real-Time Communication with WebRTC: Peer-to-Peer in the Browser*. " O'Reilly Media, Inc.".
[45] LSST. 2009. Large Synoptic Survey Telescope. http://www.lsst.org/lsst/science.
[46] Shiyong Lu and Jia Zhang. 2009. Collaborative scientific workflows. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*. IEEE, 527–534.
[47] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A Lee, Jing Tao, and Yang Zhao. 2006. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience* 18, 10 (2006), 1039–1065.
[48] Paul Luff, Jon Hindmarsh, and Christian Heath. 2000. *Workplace studies: Recovering work practice and informing system design*. Cambridge university press.
[49] Ruiqi Luo, Ping Yang, Shiyong Lu, and Mikhail Gofman. 2012. Analysis of scientific workflow provenance access control policies. In *Services Computing (SCC), 2012 IEEE Ninth International Conference on*. IEEE, 266–273.
[50] D.H. Honemann M. Robert, W.J. Evans and T.J. Balch. [n. d.]. Robert's Rules of Order. Newly Revised, 10th Edition. Perseus Publishing Company, 2000.
[51] Marta Mattoso, Claudia Werner, Guilherme Horta Travassos, Vanessa Braganholo, Eduardo Ogasawara, Daniel Oliveira, Sergio Cruz, Wallace Martinho, and Leonardo Murta. 2010. Towards supporting the life cycle of large scale scientific experiments. *International Journal of Business Process Integration and Management* 5, 1 (2010), 79–92.
[52] Ana Isabel Molina, Miguel Ángel Redondo, and Manuel Ortega. 2009. A methodological approach for user interface development of collaborative applications: A case study. *Science of Computer Programming* 74, 9 (2009), 754–776.
[53] Ana I Molina, Miguel A Redondo, Manuel Ortega, and Ulrich Hoppe. 2008. CIAM: A methodology for the development of groupware user interfaces. *J. UCS* 14, 9 (2008), 1435–1446.
[54] Golam Mostaeen, Banani Roy, Chanchal K. Roy, and Kevin A. Schneider. 2018. Fine-Grained Attribute Level Locking Scheme for Collaborative Scientific Workflow Development. In *Services Computing (SCC), 2018 IEEE International Conference on*. IEEE, 273–277.
[55] G. Mostaeen, Jeffrey Svajlenko, Banani Roy, Chanchal K. Roy, and K. Schneider. 2018. On the Use of Machine Learning Techniques Towards the Design of Cloud Based Automatic Code Clone Validation Tools. In *Source Code Analysis and Manipulation, 2018. SCAM 2018. 18th IEEE International Working Conference on*. IEEE.
[56] myExperiment. [n. d.]. Advanced FastQ manipulation. https://www.myexperiment.org/workflows/2944.html.
[57] myExperiment. [n. d.]. galaxy_101. https://www.myexperiment.org/workflows/2939.html.
[58] myExperiment. [n. d.]. NGS : Pair reads assembly with Velvet Workflow. https://www.myexperiment.org/workflows/4095.html.

[59] myExperiment. [n. d.]. Tuto Galaxy 2013 : CPB2012 - BasicProtocol3 - Calling Peaks for ChIP-seq Data. https://www.myexperiment.org/workflows/4094.html.
[60] Davide Nicolini. 2012. *Practice theory, work, and organization: An introduction.* OUP Oxford.
[61] Eduardo Ogasawara, Jonas Dias, Vitor Silva, Fernando Chirigati, Daniel Oliveira, Fabio Porto, Patrick Valduriez, and Marta Mattoso. 2013. Chiron: a parallel engine for algebraic scientific workflows. *Concurrency and Computation: Practice and Experience* 25, 16 (2013), 2327–2341.
[62] Tom Oinn, Mark Greenwood, Matthew Addis, M Nedim Alpdemir, Justin Ferris, Kevin Glover, Carole Goble, Antoon Goderis, Duncan Hull, Darren Marvin, et al. 2006. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience* 18, 10 (2006), 1067–1100.
[63] Angela Orebaugh, Gilbert Ramirez, and Jay Beale. 2006. *Wireshark & Ethereal network protocol analyzer toolkit.* Elsevier.
[64] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.
[65] Jeffrey M Perkel. 2014. Scientific writing: the online cooperative: collaborative browser-based tools aim to change the way researchers write and publish their papers. *Nature* 514, 7520 (2014), 127–129.
[66] Radu Prodan and Thomas Fahringer. 2005. Dynamic scheduling of scientific workflow applications on the grid: a case study. In *Proceedings of the 2005 ACM symposium on Applied computing.* ACM, 687–694.
[67] David Randall, Richard Harper, and Mark Rouncefield. 2007. *Fieldwork for design: theory and practice.* Springer Science & Business Media.
[68] Banani Roy and TC Nicholas Graham. 2008. An iterative framework for software architecture recovery: An experience report. In *European Conference on Software Architecture.* Springer, 210–224.
[69] Banani Roy, Amit Kumar Mondal, Chanchal K Roy, Kevin A Schneider, and Kawser Wazed. 2017. Towards a reference architecture for cloud-based plant genotyping and phenotyping analysis frameworks. In *2017 IEEE International Conference on Software Architecture (ICSA).* IEEE, 41–50.
[70] Chanchal K. Roy and James R. Cordy. 2007. A survey on software clone detection research. *Queen's School of Computing TR* 541, 115 (2007), 64–68.
[71] Chanchal K. Roy and James R. Cordy. 2008. An empirical study of function clones in open source software. In *Reverse Engineering, 2008. WCRE'08. 15th Working Conference on.* IEEE, 81–90.
[72] Chanchal K Roy and James R Cordy. 2008. NICAD: Accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization. In *Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on.* IEEE, 172–181.
[73] Gergely Sipos. 2012. Protecting the consistency of workflow applications in collaborative development environments. *Future Generation Computer Systems* 28, 3 (2012), 500–512.
[74] Gergely Sipos and Péter Kacsuk. 2009. Maintaining consistency properties of grid workflows in collaborative editing systems. In *Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on.* IEEE, 168–175.
[75] Gergely Sipos and Peter K Kacsuk. 2005. Collaborative workflow editing in the P-GRADE portal. (2005).
[76] Gergely Sipos, Gareth Lewis, Péter Kacsuk, and Vassil Alexandrov. 2005. Workflow-oriented collaborative grid portals. *Advances in Grid Computing-EGC 2005* (2005), 64–69.
[77] Apache Spark. [n. d.]. Apache Spark Lightning-fast cluster computing. https://spark.apache.org/.
[78] Chengzheng Sun. 2002. Optional and responsive fine-grain locking in Internet-based collaborative systems. *IEEE Transactions on Parallel and Distributed Systems* 13, 9 (2002), 994–1008.
[79] Chengzheng Sun and David Chen. 2002. Consistency maintenance in real-time collaborative graphics editing systems. *ACM Transactions on Computer-Human Interaction (TOCHI)* 9, 1 (2002), 1–41.
[80] David Sun and Chengzheng Sun. 2006. Operation context and context-based operational transformation. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work.* ACM, 279–288.
[81] Jeff Thomas Svajlenko et al. 2018. *Large-Scale Clone Detection and Benchmarking.* Ph.D. Dissertation. University of Saskatchewan.
[82] Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison. 2007. The triana workflow environment: Architecture and applications. *Workflows for e-Science* (2007), 320–339.
[83] Helga Thorvaldsdóttir, James T Robinson, and Jill P Mesirov. 2013. Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Briefings in bioinformatics* 14, 2 (2013), 178–192.
[84] useGalaxy. [n. d.]. An open source, web-based platform for data intensive biomedical research. https://usegalaxy.org/.
[85] Tiantian Wang, Mark Harman, Yue Jia, and Jens Krinke. 2013. Searching for better configurations: a rigorous approach to clone evaluation. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering.* ACM, 455–465.
[86] Jiachen Yang, Keisuke Hotta, Yoshiki Higo, Hiroshi Igaki, and Shinji Kusumoto. 2015. Classification model for code clones based on machine learning. *Empirical Software Engineering* 20, 4 (2015), 1095–1125.

[87] Jia Zhang. 2010. Co-Taverna: a tool supporting collaborative scientific workflows. In *Services Computing (SCC), 2010 IEEE International Conference on*. IEEE, 41–48.

[88] Jia Zhang, Daniel Kuc, and Shiyong Lu. 2014. Confucius: A tool supporting collaborative scientific workflow composition. *IEEE Transactions on Services Computing* 7, 1 (2014), 2–17.

[89] Haibin Zhu and MengChu Zhou. 2006. Role-based collaboration and its kernel mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 36, 4 (2006), 578–589.

Received February 2019; revised March 2019; accepted April 2019