Co-Taverna: A Tool Supporting Collaborative Scientific Workflows

Jia Zhang
Department of Computer Science
Northern Illinois University
DeKalb, IL USA
jiazhang@cs.niu.edu

shiyong@cs.wayne.edu

Abstract—**Scientific workflows have become an important instrument for domain scientists to synergistically integrate distributed computations and data to accelerate scientific discoveries. Existing scientific workflow tools, however, only support single scientists to compose scientific workflows in a desktop application. Nowadays, many scientific research projects are becoming increasingly larger scale, requiring that multiple research partners with different expertise collaborate from distributed organizations. Therefore, there is a critical need of a collaborative scientific workflow tool that supports domain scientists to cooperatively design, compose, annotate, execute, monitor, and manage scientific workflows over the Internet in both synchronous and asynchronous modes. This research reports the design and development of our preliminary version of a collaborative scientific workflow tool based on an open-source, single-user tool Taverna. We present our study of the role-organization-based access control technique over collaborative scientific workflow composition.**

Keywords-collaborative scientific workflows; Taverna.

I. INTRODUCTION

Modern science has yielded terabytes of heterogeneous data and a variety of data analysis and manipulation methods and tools. These resources are distributed and need to be seamlessly integrated to support effective scientific explorations. Using workflows is one such way to make the scientific exploration process structured, repeatable, configurable, and reusable [1]. In contrast to business workflows that are control flow oriented and coordinate a collection of well-defined business tasks to achieve an intended business goal, scientific workflows are dataflow oriented and streamline a collection of scientific tasks to enable and accelerate unpredictable scientific discovery [2, 3]. In recent years, scientific workflows have become an important instrument for domain scientists to streamline experiments and effectively utilize local and remote computational and data resources.

A number of scientific workflow management systems (SWFMSs) have been developed to facilitate scientific workflow activities, such as Kepler [2], Taverna [4], Triana [5], VisTrails [6], Pegasus [3], Swift [7], and VIEW [8, 9]. These tools, however, only support individual scientists to compose scientific workflows upon an installed desktop application. Nowadays, increasingly more scientific research projects have become large scale, requiring multiple research partners collaborate from distributed organizations and locations. For example, the Cancer Biomedical Informatics Grid (caBIG) initiative launched by the National Cancer Institute aims to connect the entire cancer community together to accelerate global cancer research [10]. Meanwhile, numerous researchers from a variety of domains expect to adopt various channels, including the Internet, to communicate and collaborate toward the ultimate goal.

Such large-scale, distributed scientific collaborations require collaborative scientific workflow as the underlying support, as we defined in [1]: "the computerized facilitation or automation of a scientific process, in whole or part, which streamlines and integrates people, datasets, and scientific tasks with data channels, dataflow constructs, and collaboration patterns to automate collaborative data computation and analysis for enabling and accelerating scientific discovery."

Collaborative scientific workflow poses significant challenges that cannot be handled by existing single user-oriented SWFMs lacking collaboration support and interoperability between SWFMSs [1]. Thus, there is a compelling need of a tool that supports domain scientists to cooperatively design, compose, execute, monitor, provenance track, and manage scientific workflows over the Internet in both a synchronous and an asynchronous mode.

As the first step, in this research, we focus on building a tool supporting collaborative scientific workflow composition both synchronously. In this paper, we report our design and development toward building such a collaborative scientific workflow editing tool. Without reinventing the wheel, we investigate a well-known life science-oriented scientific workflow tool, Taverna, and adapt it into a collaborative tool, called Co-Taverna, to support generic scientific collaboration. Since we focus on scientific workflows, throughout this paper, we use the terms *scientific workflows* and *workflows* interchangeably.

The remainder of the paper is organized as follows.

Section 2 surveys related work. Section 3 explains the motivation of our research project. Section 4 reports our design and development of Co-Taverna. Section 5 presents discussions. Section 6 makes conclusions.

II. Related Work

To date, several scientific workflow management systems (SWFMSs) have been developed as single-user environments, which run on local desktop computers to help individual scientists construct scientific workflows from available resources. Reported projects include Kepler [2], Taverna [4], Triana [5], VisTrails [6], Pegasus [3], Swift [7], and VIEW [8, 9].

Kepler [2] is a Java-based open-source SWFMS, where a scientific workflow is composed of uniformed components called *actors* and its execution is controlled by a dedicated computational model controller called *director*. Taverna [4] is an open-source SWFMS targeted for life science. Taverna adopts an XML-based workflow language called *SCUFL* to support workflow representation, each component being either a Web service or a Java Beanshell script-based processor supporting various bioinformatics data analysis and transformation. Triana [5] provides a sophisticated graphical user interface supporting workflow composition and modification activities, including grouping, editing, and zooming functions. VisTrails [6] focuses on workflow visualizations supporting provenance tracking of workflow evolution in addition to data product derivation history. Pegasus [3] provides a framework that maps complex scientific workflows onto distributed Grid resources. Artificial intelligence planning techniques are used for guiding workflow composition. Swift [7] combines a scripting language called *SwiftScript* with a powerful runtime system to support workflow specification and execution of large loosely coupled computations over the Grid environments. VIEW [8, 9] provides a tool that allows domain scientists to compose a scientific workflow from available resources and services. The system is featured with efficient provenance management utilizing the power of relational databases.

Each of these SWFMSs provides a platform to support individual scientists in composing scientific workflows from various resources. Their foundations center on scientific workflow models and provenance models.

Some systems show some collaboration features, in the sense that they allow a scientist to compose a scientific workflow from shared resources and services, e.g., published Grid services. However, they provide limited support for multiple scientists to collaboratively compose and manipulate a shared scientific workflow. They do not address and support user interaction and cooperation that are required and essential for an effective and efficient scientific collaboration [1].

Some SWFMSs, such as Taverna [4], declare that they support collaborative scientific workflow composition. Researchers can publish their composed scientific workflows in a dedicated social workflow space (e.g., MyExperiment [11]); others using the same SWFMS can download the workflows, make changes, and upload the new version into MyExperiment to initiate further interactions. However, such SWFMSs do not support real-time shared scientific workflow editing.

The business world recently recognizes the need of involving humans into business workflows and has developed a preliminary model [12]. However, the model is inapplicable to collaborative scientific workflows due to the fundamental differences between business workflows and scientific workflows. While business workflows are control flow oriented, scientific workflows are dataflow oriented. Furthermore, provenance data management for the reproducibility of scientific results is essential for scientific workflows but not for business workflows. Hence, scientific workflows pose a different set of requirements [13].

We studied the state of the art of the field of scientific workflows towards the support of collaborative scientific workflows and reported our observations in [1]. We also have surveyed the literature of workflow control mechanisms in a collaborative environment in [14] and observed that the current workflow control configurations have to be predefined and remain immutable throughout the execution of a workflow. With the rapid emergence of Services Computing technology [15], a workflow may select available services (e.g., a specific data processing and analysis service) at runtime. We conclude that workflow control should be driven by demands: it should be customizable and modifiable during runtime.

III. Project Motivation

Due to its popularity, Taverna [4] has been widely used as a scientific workflow editing and execution tool in life science and Grid environment [16]. Created by the myGrid project under an open-source initiative, the Taverna workbench offers a desktop authoring environment for designing and executing scientific workflows. Taverna is driven by Freefluo, its underlying workflow enactment engine. Although originally initiated for life science, Taverna workbench can be used by other domains, such as bioinformatics, cheminformatics, astronomy, social science, and music.

As it focuses on helping life scientists build scientific workflows, Taverna provides a comprehensive set of graphical widgets, such as ports and local Java Bean widgets. These widgets provide useful building blocks to help life scientists easily build scientific workflows from various resources including both local resources and remote Web services. In other words, Taverna offers a professional interface and environment to enable and facilitate domain scientists, who are not computer scientists, in creating scientific workflows.

Taverna is associated with the myExperiment [11] website to establish a social network environment for life scientists to publish and share interesting workflows with each other. Taverna [4] users can publish their scientific workflows, mostly life science workflows, in specific formatted files in a shared space MyExperiment. Others can download the workflow files and load them into their local Taverna environment and continue to work on them. In this sense, multiple scientists can collaboratively build scientific workflows, by exchanging working versions through files in the format specific to Taverna.

Taverna provides Web services compatibility, meaning that it allows users to integrate existing Web services as components into workflows. Furthermore, Taverna is written in Java, which conforms to our open-source initiative.

However, same as other existing SWFMSs, Taverna is a single-user tool supporting individual scientists to compose a scientific workflow. It does not allow multiple domain scientists to synchronously work on a shared scientific workflow. Nevertheless, throwing away all the valuable features provided by Taverna and starting everything from scratch to build another scientific workflow tool is obviously neither efficient nor desirable. Thus, our strategy is to study the Taverna code and investigate whether it can be extended into a collaborative version. Taverna is an open-source project, whose nightly built source code is accessible from the myGrid project site. This is another reason why we selected Taverna to study the plausibility of extending it into a collaborative tool.

IV. COLLABORATION PROTOCOLS

We carefully studied the latest Taverna code (version 2.0), focusing on exploring the plausibility of extending it into a collaborative version. Our goal is to create a multi-user collaborative scientific workflow environment based on the single user-based software.

As the starting point, we examined the communication paths between Taverna instances. In other word, we aim to find a way to allow two Taverna running instances to communicate with each other. By studying Taverna code, we found that the tool is built on top of an event-based mechanism, meaning that any user action (e.g., clicking a button) triggers an action. When a user selects to save a workflow, Taverna will serialize the workflow as an XML document and save it in a file. Meanwhile, when a user selects to open such an XML file, the stored workflow will be loaded into the Taverna workbench and rendered on the screen.

In Taverna, all workflows are stored in an XML-based specification language [17]. Below is a segment of XScufl code:

An XScufl file contains one workflow, with a required version number and an optional author name. The workflow comprises one to many dataflows, each comprising a sequence of elements including: name, a set of input ports, a set of output ports, a list of processors (tasks), some conditions, a set of data links (edges), and annotations.

```
<schema>
 <complexType name="Workflow">
  <sequence>
   <element name="dataflow" type="tav:Dataflow"
maxOccurs="unbounded" minOccurs="1"/>
  </sequence>
  <attribute use="required" name="version" type="tav:Version1"/>
  <attribute name="producedBy" type="string" use="optional"/>
 </complexType>

 <complexType name="Dataflow">
  <sequence>
   <element name="name" type="string"/>
   <element name="inputPorts"
type="tav:AnnotatedGranularDepthPorts"/>
   <element name="outputPorts" type="tav:Ports"/>
   <element name="processors" type="tav:Processors"/>
   <element name="conditions" type="tav:Conditions"/>
   <element name="datalinks" type="tav:Datalinks"/>
   <element name="annotations" type="tav:Annotations"
maxOccurs="1" minOccurs="0"/>
  </sequence>
 </complexType>
</schema>
```

A. *Basic collaboration model*

We thus utilize such a file system-based workflow storage mechanism to enable communication between two Taverna versions. In more detail, when a collaborator who has the write privilege saves a workflow, its serialized XML document can be propagated to another site where another collaborator has read privilege. An automatic file open action will render the same workflow on the reader's screen.

We adopt the observer design pattern [18] to build a preliminary infrastructure enabling collaboration between multiple Taverna versions. The observer pattern is a subset of the asynchronous publish/subscribe design pattern. A special subject is used to maintain a list of its dependents (observers) and automatically notify them of any state changes. Fig. 1 shows such a client/server-based infrastructure. A central server is established to support multiple Taverna users in collaborating. It acts as the subject

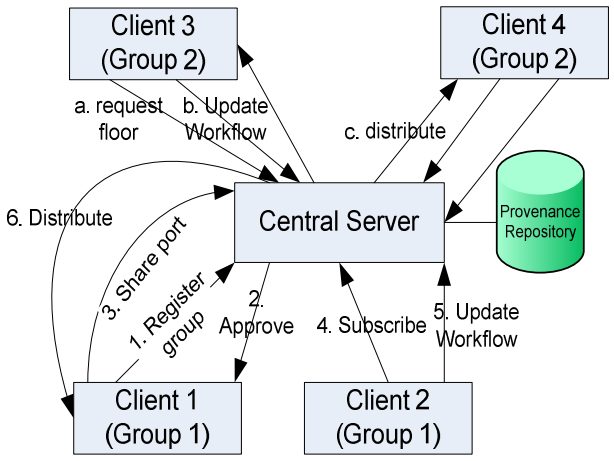

Figure 1. Collaboration protocol.

and maintains all collaborators' information. All collaborators act as observers. As shown in Fig. 1, the central server also stores and manages all provenance data, so that late comers can view shared workflows.

Fig. 1 shows possible flow scenarios. Client 1 registers a collaboration Group 1 on the central server (Step 1). Upon approval (Step 2), Client 1 shares a port to his/her potential collaborators (Step 3). Users from the invitation list (e.g., Client 2) may subscribe (Step 4) to the registered collaboration group (i.e., Group 1) and start to update the shared workflows within the group (Step 5). Any updated version will be stored in the central server and automatically distributed to all collaborators in the collaboration group (Step 6).

Currently, any action in an original Taverna workbench (including adding an element, deleting an element, updating an element, and saving a workflow) will trigger an automatic "save" action. The entire workflow will be serialized and saved to a file, which will be in turn automatically sent to the server. The server then retrieves all group members' information and delivers the up-to-date workflow file to all collaborators.

B. Advanced collaboration model

Scientific collaborations usually last for a long period of time, such as months and years. In addition, temporary discussion groups and sessions may be formed in the lifecycle of a long-term scientific collaboration process. Therefore, we constructed a hierarchical structure for the central server. In our infrastructure, a central server may host multiple collaboration groups, which may or may not have nesting relationships between them. It maintains all collaboration group information and acts as the subject for all registered collaboration groups. All observers (collaborators) are organized into corresponding collaboration groups. As shown in Fig. 1, the central server also stores and manages all provenance data, so that it becomes a repository of workflow products and decides scalability.

Within a collaboration group, a straightforward way is to allow everyone to do anything on a workflow at any time, and distribute the results to everyone in the same group. In the real life, however, typically only one person is allowed to speak at a certain moment in a group [19]. Thus, we should grant some access control policies so that only one person at a time can modify the shared workflow products and distribute the changes in the group.

We adopt the floor control technique from an extensively tested and well proved human communication protocol, Robert's Rules of Order (RRO) [19], where a single floor is maintained in a shared meeting environment. Each member requests and competes for the floor, and only the person who obtains the floor can talk in the meeting. Applying RRO to our collaboration environment, each member in a collaboration group may request a floor to gain the write privilege of the shared workflow products in the group. A simple role-based model is adopted. The person who registers a collaboration group at the central server becomes the moderator of the group, and will automatically have the control over the floor. Without losing generality, here we only allow one single floor in our project. Multiple floor-based access control facility can be realized in our future work. As shown in Fig. 1, Client 3 in Group 2 requests the floor (Step a). Upon approval, Client 3 may update the workflow (Step b) and the changes will be distributed to other collaborators in the same group (e.g., Client 4) instantaneously.

Each collaborator shall request the token first, before making any modifications on the workflow product. Otherwise, the changes will be kept locally and will not be distributed to other collaborators. The following pseudo code realizes an algorithm of the floor granting process.

Algorithm 1: Floor Granting Algorithm
Input: A collaborator releases a floor
Requirements: Release a floor.
1: Check the waiting list.
2: **if** waiting list ≠ empty **then**
3: get the top requestor of the waiting list
4: floor owner ← requestor
5: notify all members
6: remove the top requestor from waiting list
7: **else if** waiting list is empty **then**
8: floor flag ← unoccupied
9: notify all members
10:**endif**

If the floor is not occupied, the requestor will be granted the floor exclusively; otherwise, the requestor will be put into the corresponding waiting list and wait for the floor. Upon releasing a floor, the requestor on the top of the waiting list will be automatically informed and granted the floor. If there is no one in the waiting list, then nothing will happen. The following pseudo code shows the algorithm of the floor releasing process.

A moderator may force take the floor from a collaborator under certain circumstances, for example, if the collaborator loses her Internet connection.

Algorithm 2: Floor Releasing Algorithm
Input: A requestor requests a floor
Requirements: Decide whether a floor should be granted.
1: Check floor flag.
2: **if** floor ≠ taken, **then**
3: floor flag ← occupied
4: floor owner ← requestor
5: notify all members
6: return *true*
7: **else if** floor = taken **then**
8: add requestor to waiting list
9: return *false*
10: **endif**

C. *Light-weight collaboration model*

The aforementioned client/server model is based on a centralized server with the ability of permanent provenance storage. In contrast to such a formal collaboration mode, some researchers may prefer a more informal collaboration mode at some points in a specific scientific collaborative project. Informal collaborations are also called backdoor communications, implying that collaborations occur among some team members in a free and private manner. In a large-scale research project, it is common that several scientists may tend to conduct some backdoor discussions among them from time to time. In addition to free text conversations that can be supported by applications like instant messengers (IMs, which is what we plan to integrate into Taverna in the future), here we focus on sharing temporary workflow changes among a subset of collaborators.

The intermediate workflow changes of a backdoor collaboration will not be distributed to other team members instantaneously and will not be stored permanently on the central server. Instead, the changes will be shown only on the screens of the participants who join the backdoor communications, usually through invitation. The products of such backdoor communications will only be stored temporarily on each participant's local machine, unless the initiator of the communication decides to explicitly store them. Typically, any scientist in a collaboration group may decide to initiate a backdoor collaboration by sending invitations to other collaborators. More invitations may be sent in the middle of the backdoor collaboration. Such a form of backdoor communication enables peer-to-peer (P2P) communication among Taverna instances, without a central server equipped with a centralized provenance management facility.

To realize the backdoor collaboration, a straightforward way is to adopt the traditional P2P mode, where each peer (i.e., Taverna instance) is equally weighted and is enabled to communicate with each other. This implies that each Taverna instance becomes heavy-weight, meaning that we have to physically embed P2P communication code into each Taverna instance. Recall that our centralized client/server model of Co-Taverna intentionally keeps each Taverna client light weighted. This heavy-weight Taverna client requirement will constraint the reusability and flexibility of the code of the Taverna instance. In addition, our server-based communication mode is not reusable in this option.

To overcome these limitations, our solution proposes a light-weight server model. In contrast to the heavy server model discussed in the previous section, we enable a light-weight server that only serves backdoor communications and is not equipped with the ability of automatic provenance backtracking and management.

The only capability of a light-weight server is to temporarily store the latest version of the workflow product and broadcast it to all participating clients (i.e., Taverna instances). We modularize the light server as a pluggable component to the original Taverna instance code. This implies that such a server can run on every client side, in addition to the light-weight client. Between two peers who intend to communicate, only one peer has to initiate a light-weight server.

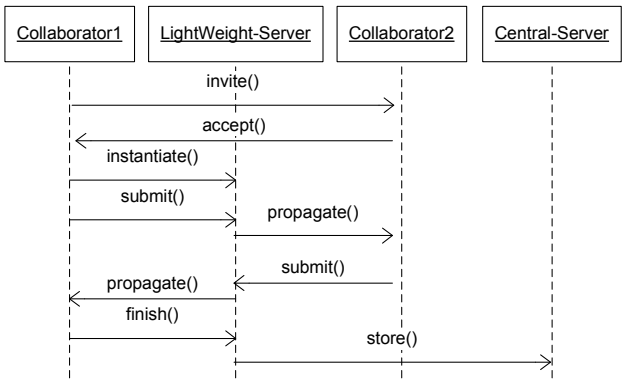

Figure 2. Light-weight collaboration protocol.

Fig. 2 illustrates the light-weight communication protocol between two collaborators using a UML sequence diagram. When one user (Collaborator1) wants to start a back-door channel, she sends an invitation (to Collaborator2). Upon receiving an agreement, the user (Collaborator1) implicitly instantiates a light-weight server at the user side and it starts to run. A signal is sent to the other party as well. It is in a light-weight mode, in the sense that it does not permanently store workflow products. As shown in Fig. 2, when Collaborator1 makes some changes to the shared workflow, the changes will be submitted to the light-weight server. The light-weight server will in turn propagate the changes to the participating Collaborator2. Similarly, when Collaborator2 makes changes, they will be propagated to Collaborator1 through the light-weight server. Finally, when the initiator (Collaborator1) decides to finish the backdoor communication, the final version of the changes can be sent to the central server to store, if so desired.

V. CO-TAVERNA 1.X

A major technical challenge is how to embed our floor-based workflow collaboration mechanism into Taverna workbench. In addition to code changes, graphical interfaces should also be adjusted as well. For example, new menus, menu items, and hot keys should be added to allow collaborative scientific workflow composition. In this section, we will discuss our strategy and solutions.

A. *System implementations*

We have successfully created a preliminary collaborative version of Taverna, called Co-Taverna 1.0, as shown in Fig. 3. Using a typical client/server model, multiple scientists

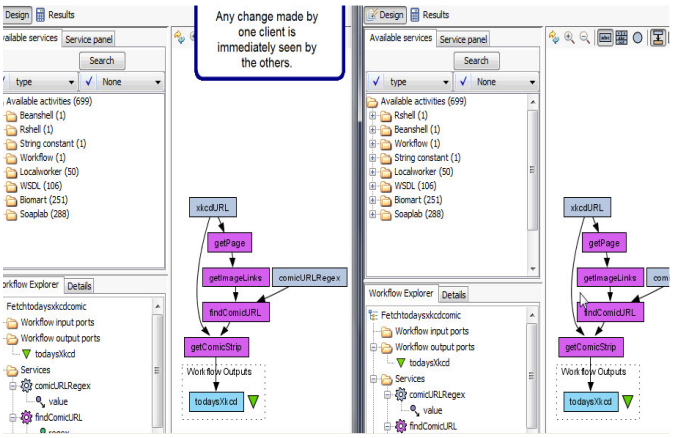

Figure 3. Collaborative Taverna 1.0.

may join in a shared scientific workflow composition session. To ease discussions, Fig. 3 shows two screens, left and right, representing two scientists working at individual screens. Both screens are running our Co-Taverna 1.0. Any change (adding or removal of components) in the shared workflow made by one scientist will be immediately shown on all collaborators' screens. Shared workflow product is stored at the server, so other collaborators may join the collaboration at any time and review the current workflow if proprietary access control allows.

Fig. 4 shows portions of the demos of Co-Taverna 1.1, where we have realized role-based P2P collaboration using the centralized server mode, as discussed in Section 4.2. As shown in the upper part of Fig. 4, we added a menu item group "Collaboration" in the menu bar, which supports five actions regarding P2P collaboration: (1) share workflow (a coordinator initiates a shared scientific workflow document), (2) connect (the coordinator allows identified participants to join), (3) disconnect (the coordinator removes a participant from the collaboration), (4) request token (request a floor to have write access), and (5) release token (release the write access of the shared workflow).

The left screen in Fig. 4 shows a scientist who starts a collaboration session. Once the scientist clicks to "Share Workflow," the collaboration can begin. As highlighted in Fig. 4, the initiator of the collaboration automatically obtains the token (floor), shown in green. He/she can also click "Release Token" to release the token; and his/her status will turn back into red by doing so. After a collaboration session is started, other scientists (upon invitations) will be able to select the "Connect" menu item to join the collaboration, and will instantaneously view the same workflow shown on the token holder's screen. As shown in Fig. 4, any collaborator can click "Request Token" to ask for the write privilege. If available, the token will be granted to the requestor.

Fig. 5 shows a portion of a screen shot illustrating that a backdoor communication is initiated between two Co-Taverna 1.1 instances. A user identifies a specific IP address (i.e., 127.0.0.1) to invite a team member to start a backdoor communication. Our current version offers six functions supporting backdoor communication, as shown in the drop down menu at the upper right corner of Fig. 4: (1) backdoor connection (initiate a backdoor communication session), (2) share workflow (manually enable workflow sharing between backdoor communication participants), (3) connect (invite an additional participant to the backdoor communication), (4) disconnect (remove a participant from the backdoor communication), (5) request token (a participant asks for the mutual exclusive floor for writing access to the shared workflow), and (6) release token (a participant releases the floor to allow other participants to request the floor).

B. *Discussions*

We realize there is much space to enhance Co-Taverna

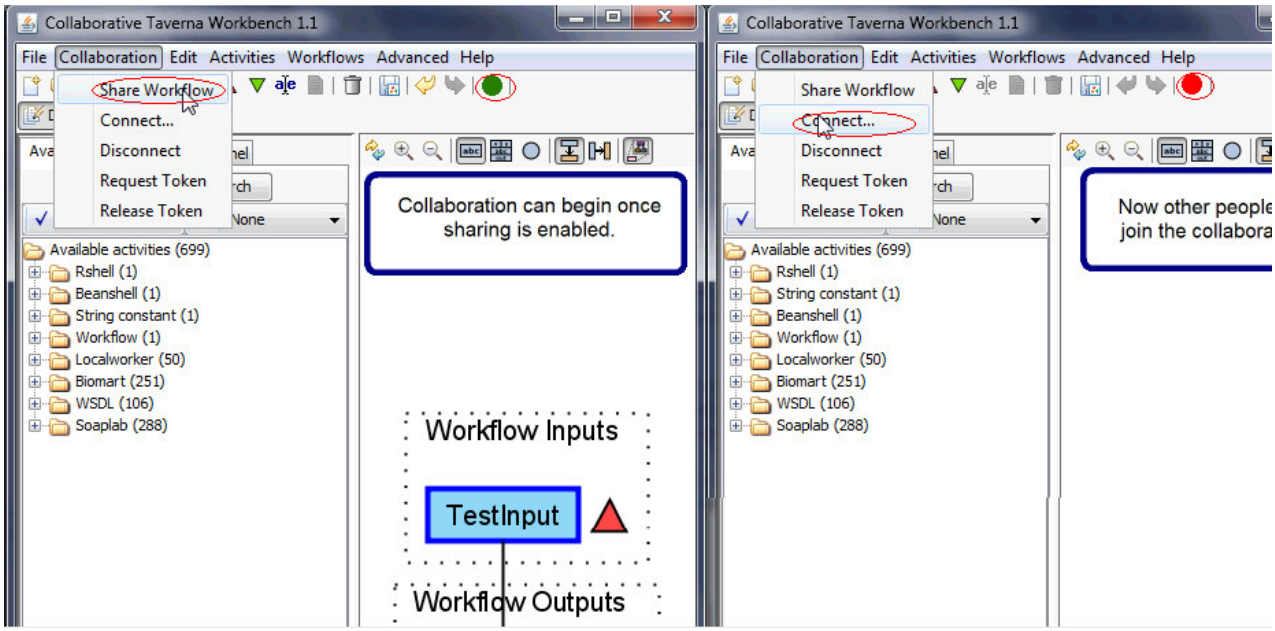

Figure. 4. P2P collaboration.

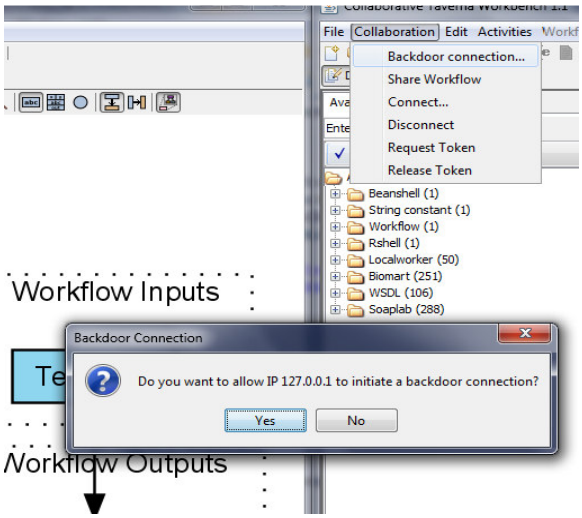

Figure. 5. Backdoor collaboration settings.

1.x for higher performance. In this section, we will briefly discuss the findings from our experience of using our preliminary tool.

For Co-Taverna to support various scales of collaborative scientific workflow composition, storage concerns have to be addressed. Since we utilize the file saving function of Taverna 2.0 to enable real-time workflow sharing, Co-Taverna 1.1 stores the entire workflow documents onto the server. Each time if a user decides to load a workflow, the entire workflow document has to be downloaded from the server and loaded into the local Taverna workbench. When a workflow becomes complicated and comprises sub-workflows, this strategy may generate significant network traffic and affect workflow retrieval and display performance. In the future, we will explore to store only workflow actions on the server. A workflow diagram will be dynamically rendered locally based on requests.

Version control is not enabled at the moment; only the latest version of a workflow product is stored and becomes accessible to collaborators. This limitation is inherited from the current Taverna code. In the future, we plan to study how to incorporate an effective and efficient version control mechanism, so that provenance management can be enabled and latecomers can view the history of workflow development.

In the current version, ownership is not granted. A rather straightforward collaboration protocol is adopted, as every collaborator competes for the floor (i.e., token) for the writing access to the entire shared workflow. In the real world, however, some portions of a workflow may be owned by a specific collaborator and others do not have direct access. For example, a particular data handling process (sub-workflow) may have to be operated by a specific scientist. In the future, we plan to establish a more comprehensive collaboration protocol to enable project-specific ownership management.

We also plan to adopt the Web services technology to restructure the system implementation of Co-Taverna 1.x. Web services is by far the best enabling technology to realize Service Oriented Architecture (SOA) for higher reusability, among many other promising features. [16]. A Web service is a programmable Web component with a standard interface and is universally accessible using standard network protocols. We plan to apply the Web services technology to refactor the entire system implementation of Co-Taverna to enable higher code reusability for later versions.

Same as traditional Taverna, each user must run a copy of Co-Taverna on the local machine. This is another reason why we plan to incorporate the Web services technology. We aim to build a collaborative scientific workflow editing service, so that users who have Internet access can use Web browsers to collaboratively compose, edit, and manage scientific workflows.

To support our incremental design, development, and testing of Co-Taverna implementations, we plan to adopt a client/server model for the project. We will establish a central server managing all workflow and provenance data as well as collaboration coordination. To allow individual scientists find available workflows, we also plan to implement a workflow publishing and discovery engine at the central server.

In spite of the aforementioned limitations that will be addressed in our future research, our Co-Taverna 1.x successfully proves the feasibility of our strategy, to extend such an existing popular scientific workflow tool into a collaborative version to support both synchronous and asynchronous scientific collaboration.

VI. CONCLUSIONS

In this paper, we report our on-going efforts of extending Taverna from a single-user version into a multi-user version, Co-Taverna. We have successfully created an initial version of Co-Taverna supporting multiple scientists in editing a shared scientific workflow. To our best knowledge, this is the first tool prototype supporting collaborative scientific workflow composition. We have designed and integrated a role-based collaboration protocol and technique and have integrated it into Co-Taverna to enable regulated scientific collaboration, based on our previous research on Internet-based computer-supported collaborative work [18, 19]. Results of our research will particularly facilitate large-scale and cross-disciplinary research projects that are collaborative in nature and require intensive user interaction from multiple distributed domain scientists.

VII. ACKNOWLEDGEMENT

This work is supported by National Science Foundation, under grant NSF0959215. The author also thanks Matthew Leverton for system implementation.

VIII. REFERENCES

[1] S. Lu and J. Zhang, "Collaborative Scientific Workflows", in Proceedings of *IEEE International Conference on Web Services (ICWS)*, Jul. 6-10, 2009, Los Angeles, CA, USA, pp. 527-534.

[2] B. Lud¨ascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the Kepler system", *Concurrency and Computation: Practice and Experience*, 2006, 18(10): pp. 1039-1065.

[3] Y. Gil, E. Deelman, J. Blythe, C. Kesselman, and H. Tangmunarunkit, "Artificial Intelligence and Grids: Workflow Planning and Beyond", *IEEE Intelligent Systems*, Jan.-Feb., 2004, 19(1): pp. 26–33.

[4] T. Oinn, M. Greenwood, M. Addis, M.N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M.R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: Lessons in Creating a Workflow Environment for the Life Sciences", *Concurrency and Computation: Practice & Experience*, 2006, 18(10): pp. 1067–1100.

[5] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, and I. Wang, "Programming Scientific and Distributed Workflow with Triana Services", *Concurrency and Computation: Practice & Experience*, 2006, 18(10): pp. 1021–1037.

[6] J. Freire, C.T. Silva, S.P. Callahan, E. Santos, and C.E. Scheidegger, "Managing Rapidly-Evolving Scientific Workflows ", *Lecture Notes in Computer Science*, May, 2006, 4145/2006: pp. 10–18.

[7] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. Laszewski, V. Nefedova, I. Raicu, T. Stef-Praun, and M. Wilde, "Swift: Fast, Reliable, Loosely Coupled Parallel Computation", in Proceedings of *IEEE International Workshop on Scientific Workflows*, Jul. 9-13, 2007, Salt Lake City, UT, USA, pp. 199–206.

[8] A. Chebotko, C. Lin, X. Fei, Z. Lai, S. Lu, J. Hua, and F. Fotouhi, "VIEW: A Visual Scientific Workflow Management System", in Proceedings of *the 1st IEEE International Workshop on Scientific Workflows*, Jul., 2007, Salt Lake City, UT, USA, pp. 207–208.

[9] C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua, and F. Fotouhi, "Service-Oriented Architecture for VIEW: A Visual Scientific Workflow Management System", in Proceedings of *the IEEE 2008 International Conference on Services Computing (SCC)*, Jul. 9-11, 2008, Honolulu, HI, USA, pp. 335-342.

[10] NCI, "Cancer Biomedical Informatics Grid (caBIG)", Available from: https://cabig.nci.nih.gov/.

[11] D.D. Roure, C. Goble, and R. Stevens, "The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows", *Future Generation Computer Systems*, 2009, 25: pp. 561-567.

[12] A. Agrawal, M. Amend, M. Das, M. Ford, C. Keller, M. Kloppmann, D. König, F. Leymann, R. Müller, K. Plösser, R. Rangaswamy, A. Rickayzen, M. Rowley, P. Schmidt, I. Trickovic, A. Yiu, and M. Zeller, "WS-BPEL Extension for People (BPEL4People), Version 1.0", Jun., 2007, Available from: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/BPEL4People_v1.pdf.

[13] E. Deelman and Y. Gil, "NSF Workshop on the Challenges of Scientific Workflows", May 1-2, 2006.

[14] C.K. Chang, J. Zhang, and K.H. Chang, "Survey of Computer Supported Business Collaboration in Support of Business Processes", *International Journal of Business Process Integration and Management (IJBPIM)*, 2006, 1(2): pp. 76-100.

[15] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*. Springer, 2007.

[16] T. Oinn, P. Li, D.B. Kell, C. Goble, A. Goderis, M. Greenwood, D. Hull, R. Stevens, D. Turi, and J. Zhao, *Taverna/myGrid: Aligning a Workflow System with the Life Sciences Community*, Workflows for E-science: Scientific Workflows for Grids (I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, eds.). Springer-Verlag, 2007, 300-319.

[17] T. Oinn, "XScufl Language Reference", 2004, Available from: http://www.ebi.ac.uk/~tmo/mygrid/XScuflSpecification.html.

[18] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, Boston, MA, USA, 1995.

[19] M. Robert, W.J. Evans, D.H. Honemann, and T.J. Balch, *Robert's Rules of Order, Newly Revised, 10th Edition*. Perseus Publishing Company, 2000.