

# Service-Oriented Architecture for VIEW: a Visual Scientific Workflow Management System

Cui Lin, Shiyong Lu, Zhaoqiang Lai, Artem Chebotko, Xubo Fei, Jing Hua and Farshad Fotouhi  
 Department of Computer Science, Wayne State University  
 {cuilin, shiyong, kevinlai, artem, xubo, jinghua, fotouhi} @wayne.edu

## Abstract

*Scientific workflows have recently emerged as a new paradigm for scientists to formalize and structure complex and distributed scientific processes to enable and accelerate many scientific discoveries. In contrast to business workflows, which are typically controlflow oriented, scientific workflows tend to be dataflow oriented, introducing a new set of requirements for system development. These requirements demand a new architectural design for scientific workflow management systems (SWFMSs). Although several SWFMSs have been developed that provide much experience for future research and development, a study from an architectural perspective is still missing. The main contributions of this paper are: i) based on a comprehensive survey of the literature and identification of key requirements for SWFMSs, we propose the first reference architecture for SWFMSs, ii) in compliance with the reference architecture, we further propose a service-oriented architecture for VIEW (a Visual sciEntific Workflow management system), iii) we implement VIEW to validate the feasibility of the proposed architectures, and iv) we present two case studies to showcase the applications of our VIEW system.*

## 1. Introduction

Scientific workflows have recently emerged as a new paradigm for scientists to integrate, structure, and orchestrate a wide range of local and remote heterogeneous services, such as Web services, Grid services, and P2P services into complex and distributed scientific processes to enable and accelerate many scientific discoveries [16]. A scientific workflow is a formal specification of a scientific process, which represents, streamlines, and automates the steps from dataset selection and integration, computation and analysis, to final data product presentation and visualization. A scientific workflow management system (SWFMS) is a system that supports the specification, modification, run, re-run, and monitoring of a scientific workflow using the workflow logic to control the order of executing workflow tasks. The

design of a reference architecture at an appropriate level of abstraction that addresses architectural requirements for SWFMSs is critical and challenging.

Although the reference architecture proposed by the Workflow Management Coalition (WfMC) [7] has been well adopted in the development of different business workflow management systems (BWFMSs), including the recent development of the YAWL system [17] that is aimed at exploring various workflow patterns [18]. Existing architectures for BWFMSs are not appropriate for SWFMSs since business workflows are typically controlflow oriented, while scientific workflows tend to be dataflow oriented, introducing a new set of requirements and challenges for system development, from the requirements of intensive user-interaction, customized user interface, reproducibility, high-end computing, interoperability, to heterogeneous data product, service, and application management. While several SWFMSs [11, 2, 22, 5, 14, 12] have been developed during the past few years, which provide much experience for future research and development, an architectural reference that can provide a high-level organization of subsystems and their interactions in an SWFMS is missing. The availability of such a reference architecture can provide a guidance for the architectural design of a particular SWFMS in various scientific domains.

To address this issue, i) we propose the first reference architecture for SWFMSs based on a comprehensive survey of the literature and identification of key requirements; ii) in compliance with the proposed reference architecture, we further propose a service-oriented architecture for the VIEW system. By leveraging SOA [21], VIEW consists of six loosely-coupled service components, each of which corresponds to a functional component that is identified in the reference architecture, whose functionality is exposed as a Web service; iii) we implement the VIEW system to validate the feasibility of the proposed architectures; and iv) we present two case studies to showcase our ongoing VIEW applications.

## 2. Seven key architectural requirements

Based on a comprehensive study of the workflow literature covering both SWFMSs and BWFMSs from an architectural perspective [9] and our own experience from the development of our previous VIEW prototype [4], in addition to the general requirements of scalability, reliability, extensibility, availability, and security, which are also required for a BWFMS, we identify the following seven key architectural requirements for an SWFMS:

*R1: User interaction support and user interface customizability.* In scientific workflows, scientists are often the end users to design, modify, run, re-run, and monitor scientific workflows. User-friendly graphical user interfaces and domain-specific visualization capability are crucial for scientists to successfully manage the full life cycle of scientific workflows and their various data products, and to speed up the *exploratory process* of arriving at a proper workflow design with appropriate parameter values and input datasets that lead to satisfactory scientific results. Therefore, a key architectural requirement is the flexibility of customizing the user interface for a science and engineering discipline, a scientific domain or problem, or according to an individual scientist's tastes and habits. An architecture that supports user interface customizability will greatly improve the reusability of system components for various scientific domains.

*R2: Reproducibility support.* Reproducibility is the fundamental principle of any science method. Scientific results produced from the execution of scientific workflows must be reproducible. Therefore, sufficient provenance information, including the derivation history of a data product, needs to be maintained in order to answer the following questions: What workflows or workflow steps are executed to produce this result? What parameter values are used? What input datasets have contributed to this result? What scientists' interactions are involved in producing this result? With such information, a scientific result can be reproduced in the same system or in other peer systems when necessary. Therefore, a key functional component for an SWFMS is the management of provenance metadata, from collection, representation, storage, querying, to visualization. Such a component is usually not required for a BWFMS.

*R3: Heterogeneous service and software tool integration.* Scientists often need to integrate and orchestrate a wide range of heterogeneous analytical and computational services, such as Web services, Grid services, and P2P services into a scientific workflow for solving a complex scientific problem. Third party software tools that are written in various programming languages and platforms should be easily integrated into a scientific workflow application in a plug-and-run fashion. Therefore, a key architectural requirement is the abstraction of various service and software tools as *workflow tasks* and the extensibility for future

service and software tool integration whose interfaces and communication protocols are yet unknown.

*R4: Heterogeneous data product management.* The execution of scientific workflows often produces huge amounts of data objects. These data objects can be primitive or complex values, files in different sizes and formats, database tables, or data objects of other forms. Scientists are often overwhelmed and lost in the sea of heterogeneous and potentially distributed data objects. Therefore, a key functional component for an SWFMS is the abstraction of various heterogeneous data objects as *data products* and the efficient management of data products for their storage, archival, browsing, searching, access, and visualization.

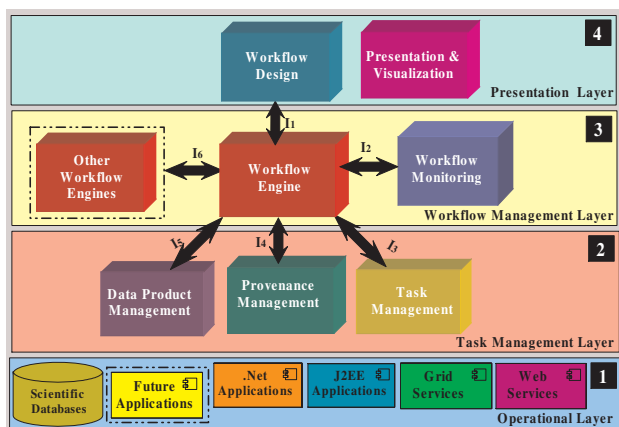
*R5: High-end computing support.* Today, many scientific problems need the support of high-end computing, such as Grid and Cluster computing. Given the fast advance of high-end computing technology, a key architectural feature is the separation of the science-focused and technology-independent problem-solving environment from the underlying often fast advance of high-end computing infrastructure. In this way, scientists can focus on their science while utilizing the state-of-the-art computing facilities behind the scene.

*R6: Workflow monitoring and failure handling.* The monitoring of the progress of the execution of scientific workflows is very important for scientists, particularly for long-running scientific workflows. Moreover, since scientific workflows are designed and modified by scientists in an ad hoc fashion and can involve various services that are accessed over network communications, many exceptions or failures can occur in an unforeseeable way. Finally, the complexity and scale of data analysis and computation in scientific workflows impose additional challenges on workflow monitoring and failure handling.

*R7: Interoperability.* As more and more scientific research projects become collaborative in nature and involve several geographically distributed organizations, many scientific workflows are distributed and collaborative, consisting of several subworkflows, each of which is managed by a different SWFMS. Therefore, it is very important that various SWFMSs can interoperate with one another to take advantage of the software tool library and salient features provided by each SWFMS. Moreover, the interoperability of subsystems will enable the reuse of a subsystem in another SWFMS.

## 3. A reference architecture for SWFMSs

Although the reference architecture proposed for BWFMSs by WfMC has been widely accepted, this reference architecture does not satisfy the key requirements from R1 to R5 for an SWFMS. Recent development of several SWFMSs, including our previous VIEW prototype [4], provide much experience and insight to the design of the



**Figure 1.** A reference architecture for SWFMSs.

reference architecture shown in Figure 1, which consists of four logical layers, seven major functional subsystems, and six interfaces.

**Layers.** The first layer is the *Operational Layer*, which consists of a wide range of local or remote heterogeneous data and services, software tools, and their operational environments, including high-end computing environment. The separation of the Operational Layer from other layers isolates high-end computing environment from higher-level functionalities and thus satisfies R5.

The second layer is called the *Task Management Layer*, which abstracts underlying heterogeneous data into data products, services and software tools into workflow tasks, and provides efficient management for data products, workflow tasks, and provenance metadata. Therefore, the Task Management Layer satisfies requirements R2, R3, and R4. Moreover, the separation of the Task Management Layer from the Operational Layer promotes the extensibility of the Operational Layer with new services and new high-end computing facilities, and localizes system evolution due to hardware or software advances to the interfaces between the Operational Layer and the Task Management layer.

The third layer is the *Workflow Management Layer*, which is responsible for the execution and monitoring of scientific workflows. At this layer, the building blocks of a scientific workflow are the workflow tasks provided by the underlying Task Management Layer. Moreover, the interoperability issues between the workflow engine and other workflows engine is addressed in this layer. Therefore, the Workflow Management Layer addresses requirements R6 and R7. Finally, the separation of the Workflow Management Layer from the Task Management Layer not only isolates the choice of a workflow model from the choice of a task model, provenance model, and data product model, but also separates the enforcement of workflow logic from the execution of the constituent workflow tasks and the creation

of data products.

The fourth layer is the *Presentation Layer*, which provides the functionality of workflow design and various user interfaces and visualizations for all assets of the whole system. The *Presentation Layer* has interfaces to each lower layer (not shown in the figure for simplicity). The separation of the *Presentation Layer* from other layers provides the flexibility of customizing the user interfaces of the system and promote reusability of the rest of system components for different scientific domains. Thus, this separation supports architectural requirement R1.

**Subsystems.** The seven major functional subsystems correspond to the key functionalities required for an SWFMS. The subsystem of *Workflow Design* is responsible for the design and modification of scientific workflows. Several complementary modes of environments can be provided for the design and modification of workflows, including graphical tools and scripting languages either from a standalone software designer or from a web portal tool. The subsystem of *Presentation and Visualization* is very important for complex data analysis applications in which the presentation of workflows and visualization of various data products and provenance metadata are the key to gain insights and knowledge from large amount of data and metadata. These two subsystems are located at the *Presentation Layer* to meet requirement R1. The subsystem of *Workflow Engine* is at the heart of the whole system and is a subsystem that is responsible for the run and re-run of scientific workflows, using the workflow logic to control the order of initiating workflow tasks. The subsystem of *Workflow Monitoring* meets requirement R6 and is in charge of monitoring the status of workflow execution and if failures occur, providing tools for failure recovery. The subsystem of *Task Management* is responsible for the registration, annotation, searching, and execution of workflow tasks. The subsystem of *Provenance Management* meets requirement R2 and is mainly responsible for the management of scientific workflow provenance metadata, including their representation, storage, archival, searching, and visualization. The subsystem of *Data Product Management* meets requirement R4 and is mainly responsible for the management of heterogeneous data products. One key challenge for data product management is the heterogeneous and potentially distributed nature of data products, making efficient access and movement of data products an important research problem.

**Interfaces.** Only six interfaces are explicitly defined, which show how the Workflow Engine interacts with other subsystems. *Interface I<sub>1</sub>* isolates the workflow design environment from workflow execution environment, and provides APIs for the execution of workflow specifications that are provided by different workflow design tools. *Interface I<sub>2</sub>* isolates the execution of a workflow from its monitoring and failure handling. *Interface I<sub>3</sub>* separates workflow

scheduling from individual task execution. *Interface  $I_4$*  is used for provenance tracking and reproducibility support. *Interface  $I_5$*  separates workflow management from data product management. Finally, *Interface  $I_6$*  defines the APIs for interoperating with other workflow engines provided by various vendors, corresponding to requirement R7.

#### 4. Service-Oriented Architecture for VIEW

In order to validate the feasibility of our proposed reference architecture, we firstly present the advantages of using SOA in SWFMSs; secondly, we propose a service-oriented architecture for our VIEW system that complies with the reference architecture, followed by the description of each major service component; finally, we present the implementation and two case studies to showcase the applications of our VIEW system.

##### 4.1 Advantages of using SOA in SWFMSs

While the emergence of SOA as an architectural paradigm provides many benefits for distributed computing [6], we identify the following advantages of using SOA specifically for the development of an SWFMS.

1) *Service loose coupling.* Service loose coupling minimizes the dependencies among subsystems of an SWFMS by the definitions of a set of language and platform independent interfaces. In our proposed architecture, each subsystem's functionality is exposed as a Web service. As a result, an SWFMS can be developed on demand from various subsystems provided by different parties as Web services. One can also easily switch from one service to another for each subsystem. For example, there may be several provenance management services available, and using SOA, one can use and switch any provenance management service on demand for a specific SWFMS.

2) *Service abstraction and autonomy.* A Web service provides an abstract interface that is independent from its implementation. In addition, each Web service is autonomous in the sense that a service provider has the control over the application logic that the Web service encapsulates. As a result, a service provider can dynamically change the implementation and deployment environment of a Web service for a subsystem of an SWFMS with no downtime for the SWFMS as long as such changes do not affect the defined interface. Such autonomy also greatly facilitates the management of the development and evolution of the whole system.

3) *Service reusability.* As each subsystem of an SWFMS becomes a uniform computing unit with standard interface descriptions and universal accessibility through standard communication protocols. Each subsystem can be reused across various SWFMSs, even simultaneously used by both local SWFMSs and other SWFMSs across the Internet.

4) *Service discoverability.* As each subsystem of an SWFMS is implemented as a Web service that is enriched

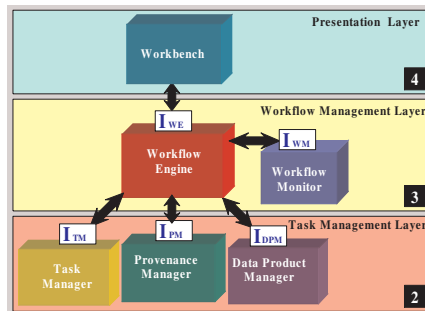


Figure 2. Overall architecture of the VIEW system.

with a semantic description, one can register the service in some public service registries. As a result, the subsystem becomes discoverable and can be selected and used by other SWFMSs on demand.

5) *Service interoperability.* Service interoperability is enabled by the open standards of messages and communication protocols for Web services, which are supported by a large body of IT industry and the Web Services Interoperability Organization (WS-I). Using Web services, the interoperability of subsystems within an SWFMS and the interoperability across various SWFMSs can be greatly improved, thus satisfying requirement R7.

##### 4.2 Overall architecture and major subsystem architectures

The overall architecture of VIEW in Figure 2 consists of six loosely-coupled, autonomous, reusable, and discoverable service components that correspond to the main functional subsystems proposed in the reference architecture. The *Operational Layer* is omitted from the figure for simplicity. Except for *Workbench*, the interface for each service component is defined and described by WSDL:  $I_{WE}$ ,  $I_{WM}$ ,  $I_{TM}$ ,  $I_{PM}$  and  $I_{DPM}$  for the interface of the Workflow Engine, Workflow Monitor, Task Manager, Provenance Manager, and Data Product Manager, respectively. Service components interact with one another by Web service invocation using SOAP messages via Internet-based protocols. The details of these interfaces are not presented due to space limit. In the following, we focus our discussion on the architectural details of major service components.

**Workbench.** The *Workbench* subsystem implements the functions of workflow design, presentation, and visualization identified at the Presentation Layer in the reference architecture. Currently it consists of five components (see Figure 3:(a)): *Workflow Designer*, *Provenance Explorer*, and the *GUIs* for Task Manager, Workflow Monitor, and Data Product Manager, respectively. The *Workflow Designer* provides a scientist-friendly GUI for the design and modification of scientific workflows. A scientist can drag and drop existing workflow tasks into the design panel and

link them one to another using various dataflow and controlflow constructs. Scientific workflows are saved in XML files into a local *Workflow Repository* as well as into the *Provenance Manager* via  $I_{PM}$ . In this way, the provenance capability can be turned off to avoid provenance tracking overhead when necessary. Each scientific workflow is stored in two XML files: a *specification file* to store the logical structure and components of the scientific workflow, and a *layout file* to store the graphical layout information of the scientific workflow. While both files are needed to display and manipulate a scientific workflow in *Workbench*, only the specification file is needed for the execution of a scientific workflow. The separation of presentation from content improves the interoperability of our workflow model and user interface customizability (requirement R1).

Provenance Explorer enables a user to browse, query, and visualize scientific workflow provenance metadata. Moreover, together with the GUI for *Data Product Manager*, one can present and visualize various data products, from simple data values and plain texts, to complex ones such as 3D multimodality brain images. The GUIs for Task Manager, Workflow Monitor and Data Product Manager provide scientist-friendly user interfaces for the management of these subsystems. They interact with subsystems via  $I_{TM}$ ,  $I_{WM}$ , and  $T_{DPM}$ , respectively. The loosely coupled nature provided by SOA improves the customizability of these user interface designs (requirement R1).

**Workflow Engine.** The architecture of *Workflow Engine* is shown in Figure 3:(c). Centered around *Scheduler*, *Workflow Engine* consists of seven functional components: *Scheduler*, *Translator*, *Controlflow Management*, *Dataflow Management*, *Workflow Status Management*, *Workflow Status Storage*, and *Provenance Collector*. There are four salient features for this architecture design. First, the *Translator* component provides a mapping scheme for translating a workflow specification into an optimized executable workflow representation. Second, the separation of controlflow and dataflow management from workflow scheduling greatly improves the extensibility of our workflow model since the introduction of additional controlflow and dataflow constructs can be achieved by upgrading their individual components without modifying other components of the service component. Third, *Workflow Status Management* and durable *Workflow Status Storage* provides a foundation for workflow monitoring and failure handling (requirement R6). Finally, *Provenance Collector* separates the concern of provenance collection and is responsible for collecting all provenance information and storing them into *Provenance Manager* via  $I_{PM}$ .

**Task Manager.** The architecture of *Task Manager* is shown in Figure 3:(b). It consists of two layers: the wrapper layer and the task layer. The wrapper layer contains various wrappers that can dynamically wrap existing ser-

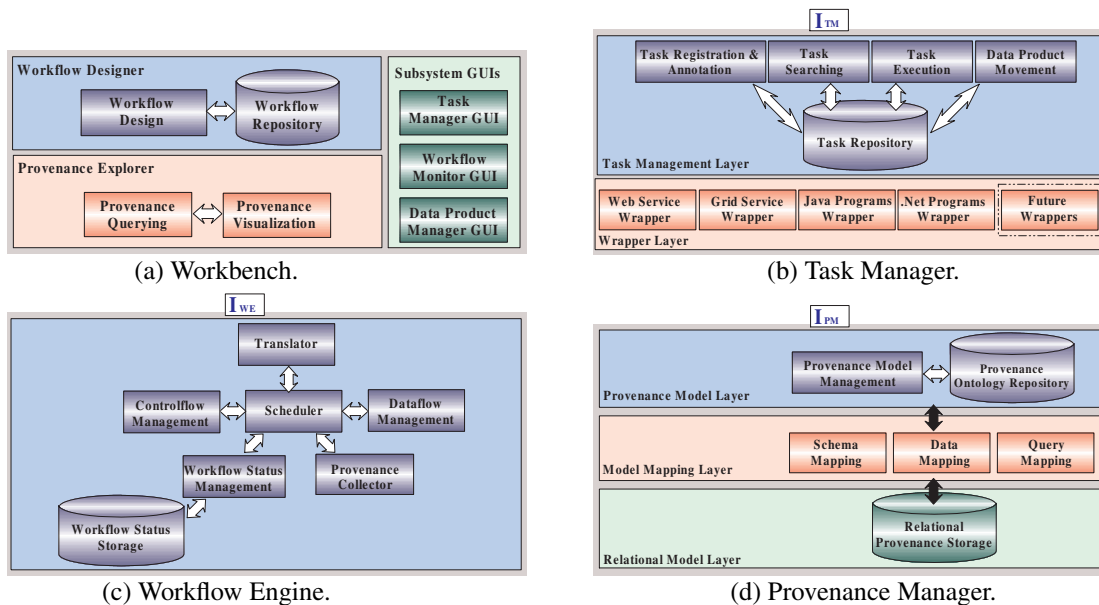
vices or software tools as workflow enabled tasks. Many of these wrappers are generic in the sense that they can be applied to a category of applications. For example, the generic Web service wrapper can be used to wrap an operation of an arbitrary Web service into a workflow task. In this way, all Web services become workflow tasks and can interoperate with other services and software tools in the framework of scientific workflows. The introduction of new categories of services or software tools is achieved by the introduction of a new wrapper. Therefore, our architecture promotes the ability of heterogeneous service and software tool integration (requirement R3). The task layer provides various functions for workflow task management, including registration, annotation, searching, execution, and data product movement. Data product movement is used for the movement of data products generated from task execution to *Data Product Manager* or the other way around.

**Data Product Manager.** Our current implementation of *Data Product Manager* is still very preliminary and includes the functions of registration, searching, and data product movement. Our vision for this subsystem is that a user and the *Workflow Engine* will access various data products transparently with respect to their heterogeneity and distribution (requirement R4). We expect our architectural design will facilitate the realization of such a vision.

**Provenance Manager.** The Provenance Manager shown in Figure 3:(d) includes three layers: the provenance model layer, the relational model layer, and the model mapping layer.

The provenance model layer contains a provenance model management component and a provenance ontology repository. This layer is responsible for the representation of scientific workflow run provenance via domain ontologies that serve as vocabularies to describe and serialize provenance metadata. Such ontologies may include general provenance vocabularies and ontologies used to represent knowledge in a particular scientific field, e.g., bioinformatics or medicine. To address the requirements of provenance representation interoperability, extensibility, and semantic integration in VIEW, we use Semantic Web technologies for provenance representation. In particular, Web Ontology Language (OWL) is used to express ontologies, Resource Description Framework (RDF) is used to serialize provenance metadata, and the RDF query language SPARQL is used to express provenance queries.

The relational model layer includes a relational provenance storage, represented by a Relational Database Management System (RDBMS), which serves as an efficient backend to store and query provenance metadata. In this layer, provenance metadata is stored in relations or tables and is queried using Structured Query Language (SQL). The requirements addressed by this layer include efficiency and scalability of provenance metadata management.



**Figure 3.** Architectures for major subsystems.

Finally, the model mapping layer serves as an integration medium between the two other layers. It includes three mappings: (1) schema mapping to generate a relational database schema based on an ontology that is used to represent provenance metadata, (2) data mapping to map provenance metadata in RDF to relational tuples and store them into the relational database, and (3) query mapping to translate provenance queries in SPARQL into relational queries in SQL that can be executed by the RDBMS. The main challenge of this layer is to provide various efficient Semantics-preserving mappings between the two different data models. More details on provenance storage and querying in VIEW are available in [3], where a sample provenance ontology is described and the three mappings are further studied and experimentally evaluated.

**Workflow Monitor.** Our current implementation of *Workflow Monitor* is still very preliminary and focus on the implementation of monitoring workflow execution status. Future implementation will introduce other features including forward recovery and backward recovery in the case of failures.

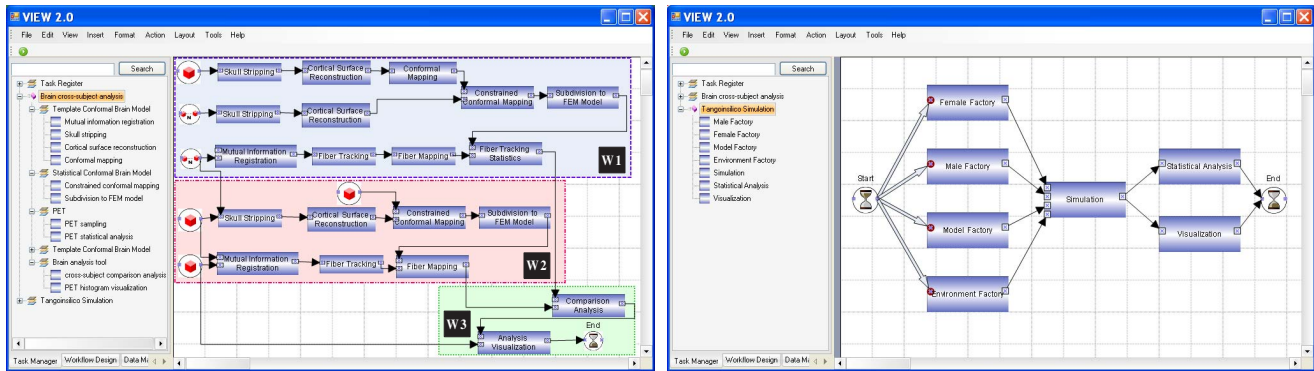
### 4.3 Implementation and case studies

We implemented the proposed service-oriented architecture in our VIEW system using Microsoft .NET 2005 that supports Web service programming. First, we reused the existing components of our previous VIEW prototype [4], which was built upon a component-based modular architecture, by upgrading original codes from Native VC++ to Managed C++, which can interoperate with all other .NET capable languages; second, we restructured the VIEW system according to the proposed architecture and introduced three new service components: Workflow Monitor, Task

Manager, and Data Product Manager; third, we implemented a set of functions that need to be exposed as Web services for each service component; fourth, we manipulated and transported XML-format data among memory, files and databases using ADO.NET and SQL Server 2005; finally, we enhanced the Workbench in its presentation and visualization capability using VTK and OpenGL. Due to space limit, we will not present the evaluation result of the system in this paper, but only comment that VIEW is sufficient to support user-interaction intensive and parameter-intensive scientific workflows as demonstrated in the following two case studies.

**Neurological disorder diagnosis.** Advanced multimodality imaging has been prevalently used for the diagnosis of brain disease. However, due to the underlying complexity in management of multimodality data and analysis tasks, it requires hours, even days, to completely analyze one patient's multimodality imaging data, which makes it impossible to employ a large number of existing datasets for population-based image analysis. We have successfully applied the VIEW system to the cross-subject brain imaging analysis for computer-assisted diagnosis of neurological disorders. Figure 4:(a) demonstrates one part of the entire workflow, which integrates many tasks and datasets involved in the cross-subject analysis of diffusion tensor imaging (DTI), thus efficiently processing and producing the final results of fiber tract analysis.

In a nutshell, this sub-workflow registers different brains in a common space, Conformal Brain Template (CBT), in order to enable cross-subject comparison. The workflow portion shaded in blue (labeled as W1) processes the normal subjects to generate a normal population-based distrib-



**Figure 4.** (a) A user-interaction intensive scientific workflow in VIEW for neurological disorder diagnosis. (b) A parameter-intensive scientific workflow in VIEW for biological simulations.

ution of the CBT, while the workflow portion shaded in pink (labeled as W2) handles an individual patient’s DTI data. After mapping fibers into the CBT, the comparison analysis shaded in green (labeled as W3) can precisely pinpoint the locations of abnormalities. Note that, users need to frequently rotate, manipulate and interact with the visual imaging data in the scientific workflow. Our VIEW system can successfully satisfy and facilitate this user-interaction intensive and visual content intensive image application analysis. Therefore, the designed workflow allows batch processing of a large population of subjects, which helps identify patients and their specific abnormal neurocortex areas.

**Biological simulations.** Pheromones can be used as attractants for the opposite sex in many environments; however, little is known about the search strategies employed in responding to pheromone in the marine environment. The spawning behavior of males of the polychaete *Nereis succinea* is known to be triggered at close range by a high concentration of pheromone released by females. Since pheromone also causes acceleration of swimming and increased turning, in addition to eliciting ejaculation, we propose the hypothesis that these behaviors elicited by low concentrations of pheromone can be used by males to find females. To test this hypothesis, we have developed a simulation scientific workflow using VIEW that is controlled by 49 parameters. The workflow is shown in Figure 4:(b), in which workflow tasks *Male Factory* and *Female Factory* are used to generate any number of different kinds of male and female worms based on the parameters specified by a user; *Model Factory* and *Environment Factory* are used to generate the experiment and environment model, respectively; task *Simulation* calculates the movement of the worms step by step with all intermediate data recorded for each step so that output of the model can be used for future analysis; finally, task *Visualization* is used to display the movement trajectories and task *Statistical Analysis* is used to analyze the simulation results. Our simulations show that complex turning behavior leading to greater chances of successful

mating encounters. Similar behavior was recorded in field observations. We are currently enhancing this workflow to determine what pheromone concentrations produce significant increases in the probability of mating encounters.

## 5. Related work

Although the term “scientific workflows” were first coined by Vouk and Singh in 1996 [19] for workflow applications in scientific problem solving environments, only recently there is an increasing momentum for the research and development of SWFMSs and their applications, due to the increasingly demanding requirements of many compute-intensive and data-intensive scientific applications, and enabled by the underlying advances of computing technologies, notably Service computing [14] and Grid computing [20]. Scientific workflows leverage existing techniques developed for business workflows but deviate from them as a result of a different set of scientific requirements raised from a wide range of science and engineering problems [15]. While business workflows are controlflow oriented with the mission of carrying out business logic to achieve a business goal, scientific workflows tend to be dataflow oriented aimed at enabling, facilitating, and speeding up the derivation of scientific results from raw datasets.

Many BWFMSs have been developed in the past two decades [8, 13, 1, 10], and many of them adopted the reference architecture proposed by the Workflow Management Coalition [7] or its variants. However, existing architectures for BWFMSs are not suitable for SWFMSs as they do not satisfy key requirements R1 to R5.

Several SWFMSs have been developed over recent years. The Kepler system [11] is a Java-based open source SWFMS. A scientific workflow is composed from components called *actors* and its execution is controlled by a computational model controller called *director*. Kepler inherits its GUI from Ptolemy II for workflow composition and modification. The Taverna system [14] is another Java-based open source SWFMS mainly targeted for life science.

Currently, Taverna supports a repository of Web services for various bioinformatics data analysis and transformation. Taverna uses an XML-based workflow language called *SCUFL* for workflow representation with each component being either a Web service or a *processor* developed using Java Beanshell script. The Triana system [12] has a sophisticated graphical user interface for workflow composition and modification, including grouping, editing, and zooming functions. Coming from the gravitational wave field, the system contains a large repository of tools for data analysis and processing. The VisTrails system [2] is developed to manage visualizations and is the first system that supports provenance tracking of workflow evolution in addition to tracking the data product derivation history. The Pegasus system [5] provides a framework which maps complex scientific workflows onto distributed grid resources. Artificial intelligence planning techniques are used in Pegasus for workflow composition. Finally, the Swift system [22] combines a novel scripting language called *SwiftScript* with a powerful runtime system to support the concise specification, and reliable and efficient execution, of large loosely coupled computations over Grid environments.

Although these systems provide much experience in future research and development, a reference architecture that can address the architectural requirements specifically for SWFMSs is missing, which is the motivation of this research.

## 6. Conclusions and future work

We proposed a reference architecture for SWFMSs and presented an SOA based design of the architecture in the VIEW system. Ongoing work includes the extension of our architecture to address security issues, a more comprehensive evaluation of the VIEW system and the development of various scientific workflow applications using VIEW.

## References

- [1] G. Alonso, R. Günthör, M. Kamath, D. Agrawal, A. Abbadi, and C. Mohan. Exotica/FMDC: A workflow management system for mobile and disconnected clients. *Distributed and Parallel Databases*, 4(3):229–247, 1996.
- [2] S. Callahan, J. Freire, E. Santos, C. Scheidegger, C. Silva, and H. Vo. VisTrails: visualization meets data management. In *SIGMOD Conference*, pages 745–747, 2006.
- [3] A. Chebotko, X. Fei, C. Lin, S. Lu, and F. Fotouhi. Storing and querying scientific workflow provenance metadata using an RDBMS. In *Proc. of the Second IEEE International Workshop on Scientific Workflows and Business Workflow Standards in e-Science*, pages 611–618, 2007.
- [4] A. Chebotko, C. Lin, X. Fei, Z. Lai, S. Lu, J. Hua, and F. Fotouhi. VIEW: a Visual Scientific Workflow Management System. In *IEEE SWF*, pages 207–208, 2007.
- [5] E. Deelman, G. Singh, M. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. Berriman, J. Good, A. Laity, J. Jacob, and D. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming Journal*, 13(3):219–237, 2005.
- [6] T. Erl. *Service-oriented architecture concepts, technology and design*. Pearson Education Inc., 2005.
- [7] D. Hollingsworth. The workflow reference model. *The Workflow Management Coalition*, 1994.
- [8] F. Leymann and D. Roller. Business process management with FlowMark. In *COMPCON*, pages 230–234, 1994.
- [9] C. Lin and S. Lu. Architectures of workflow management systems: A survey. Technical Report TR-SWR-01-2008, January 2008.
- [10] L. Liu, C. Pu, and D. Ruiz. A systematic approach to flexible specification, composition, and restructuring of workflow activities. *J. Database Manag.*, 15(1):1–40, 2004.
- [11] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.
- [12] S. Majithia, M. Shields, I. Taylor, and I. Wang. Triana: A graphical web service composition and execution toolkit. In *ICWS*, pages 514–524, 2004.
- [13] J. A. Miller, D. Palaniswami, A. P. Sheth, K. Kochut, and H. Singh. Webwork: METEOR<sub>2</sub>'s web-based workflow management system. *J. Intell. Inf. Syst.*, 10(2):185–215, 1998.
- [14] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, R. Greenwood, T. Carver, K. Glover, M. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.
- [15] I. Taylor, E. Deelman, D. Gannon, and M. Shields. *Workflows for e-science*. Springer-Verlag London Limited, 2007.
- [16] A. Tsalgatidou, G. Athanasopoulos, M. Pantazoglou, C. Pautasso, T. Heinis, R. Grønmo, H. Hoff, A. Berre, M. Glittum, and S. Topouzidou. Developing scientific workflows from heterogeneous services. *SIGMOD Record*, 35(2):22–28, 2006.
- [17] W. van der Aalst, L. Aldred, M. Dumas, and A. ter Hofstede. Design and implementation of the YAWL system. In *CAiSE*, pages 142–159, 2004.
- [18] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
- [19] M. Vouk and M. Singh. Quality of service and scientific workflows. In *Quality of Numerical Software*, pages 77–89, 1996.
- [20] J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Record*, 34(3):44–49, 2005.
- [21] L. Zhang, J. Zhang, and H. Cai. *Services Computing*. Springer, 1 edition, 2007. ISBN-10:354038281X.
- [22] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, I. Raicu, T. Stef-Praun, and M. Wilde. Swift: Fast, reliable, loosely coupled parallel computation. In *IEEE SWF*, pages 199–206, 2007.