

## **ABSTRACT**

CHANDRA, SANDEEP. Service Based Support for Scientific Workflows (Under the direction of Dr. Mladen A. Vouk)

A Problem Solving Environment (PSE) is a computer-based system that provides all the computational facilities necessary to solve a target class of problems, that is, it efficiently supports a specific set of scientific workflows. A special class of PSE's are those that rely on networks. Network-based PSEs are collections of distributed applications, interfaces, libraries, databases, programs, tools, clients, and intelligent agents, which facilitate user interaction and cooperative execution of the components charged with the solution tasks. Thus, the need for effective and efficient communication among the PSE components is obvious. This has resulted in a proliferation of communication building blocks, or middleware, for distributed scientific computing and problem solving. The most recent, and quite promising, option is the availability of network-based services, sometimes called Web Services.

This work is concerned with evaluation of the feasibility, usability and effectiveness of service-based support for scientific workflows. A successful open source proof of concept architecture and framework was developed and assessed using a Bioinformatics scientific problem solving workflow. Access to the data, computations, and user interfaces is based on the services architecture and standards such as XML for data descriptions, WSDL for service descriptions, SOAP for service delivery, and UDDI for service registration and brokering. This service-oriented approach facilitates

integration of disparate data acquisition and analysis applications that participate in complex scientific problem solving processes. The developed framework, called Scientific Data Management Service Workflow System, or SDMSWS, was found to be sufficiently flexible and versatile that it was possible to effect appliance-like composition and use of standard-conforming workflow services. An open-source solution, such as SDMSWS, enables services from different organizations, platforms and domains to interoperate seamlessly through standard interoperability protocols. This is contrasted with similar systems that implement proprietary service solutions.

In order to compare SDMSWS with other systems that support scientific problem solving workflows, it was necessary to develop a set of measures. They include measures related to end-user issues, workflow issues, services issues, networking issues, and a variety of more general issues. The selection and the definition of the metrics, and the rationale behind them, are discussed. The comparative analysis reveals that its open source nature has some distinct advantages. For example, the framework minimizes the customization and integration of new component workflow services, and it shows that, unlike commercial solutions, it is relatively independent of the service domains and can incorporate any network-based service that conforms to standard Web Services description and protocols. It is the finding of this work that complex distributed scientific workflows can be, and should be, supported using open-source service-based solutions. However, current standards, for workflow description, interchange and execution were found wanting and further work will be needed before one can depend on them in practical scientific workflow environments.

# **SERVICE BASED SUPPORT FOR SCIENTIFIC WORKFLOWS**

by  
**SANDEEP CHANDRA**

A thesis submitted to the Graduate Faculty of  
North Carolina State University  
in partial fulfillment of the  
requirements for the Degree of  
Master of Science

**COMPUTER SCIENCE**

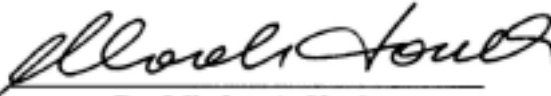
Raleigh

2002

**APPROVED BY:**

  
\_\_\_\_\_  
**Dr. Munindar P. Singh**

  
\_\_\_\_\_  
**Dr. Peter R. Wurman**

  
\_\_\_\_\_  
**Dr. Mladen A. Vouk**  
(Chair of Advisory Committee)

## **BIOGRAPHY**

Sandeep Chandra is from Hyderabad, India. He received his Bachelor's degree in computer science from BVB College, Osmania University, Hyderabad, India. He was pursuing a Postgraduate program in Master of Computer Applications from VSPGS, Osmania University, Hyderabad, India, but discontinued it upon his acceptance to the Master of Science program in the department of Computer Science at North Carolina State University.

At NC State, he was a part of the Scientific Data Management (SDM) Project, and worked on his thesis under the direction of Dr. Mladen Vouk. Also, as part of the SDM project he spent a year and a half working with scientists and biologists from Oak Ridge National Labs, Lawrence Livermore National Labs and San Diego Super Computer Center.

## **ACKNOWLEDGEMENTS**

I express my sincere gratitude to my advisor, Dr. Mladen Vouk, for his guidance and support throughout the research of this thesis. Working with him has been a fantastic experience. I also thank Dr. Peter Wurman and Dr. Munindar Singh for serving on my thesis committee. It has been a learning experience working with them as part of my course work and research interests.

I appreciate the help extended by my colleagues in the Scientific Data Management (SDM) project - Zhengang Cheng and Sangeeta Bhagwanani. Zhengang has guided me through many aspects of this research. I acknowledge members of the SDM project from San Diego Super Computer Center, Oak Ridge National Labs and Lawrence Livermore National Labs for their prompt help in many issues of the project.

My friends, both here at NC State and back home in India, have been supportive of all my endeavors – I can't thank them enough. Finally, I would sincerely like to thank my parents, Kishan and Madhu, for their love, and my brother and sister-in-law, Sumeer and Malini, for having constantly encouraged me to pursue higher studies and setting such high academic standards for me to follow.

This work has been supported in part by the DOE SciDAC grant/contract DE-FC02-01ER25484, by IBM, and by NCSU CACC.

# TABLE OF CONTENTS

List of Tables.....	vii
List of Figures.....	viii
1 Introduction.....	1
1.1 Issues.....	1
1.2 Contribution.....	3
1.3 Organization of Thesis.....	5
2 Web Service and Workflow Concepts.....	6
2.1 Web Service.....	6
2.1.1 Architecture.....	6
2.1.2 Operations.....	7
2.2 Web Service Stack.....	8
2.3 Universal Discovery, Description and Integration (UDDI).....	9
2.4 Web Service Description Language (WSDL).....	10
2.5 Simple Object Access Protocol (SOAP).....	13
2.5.1 SOAP Message.....	13
2.6 Scientific Workflows.....	14
2.6.1 Pipeline and Workflow.....	15
2.7 Web Service-Workflow Architecture.....	16
3 Scientific Data Management Service Workflow System (SDMSWS).....	18
3.1 Architecture, Working and Components.....	18
3.2 Bioinformatics Workflow Service.....	25

3.2.1 Gene-Expression Workflow.....	27
3.2.2 Performance Measure.....	30
3.3 Current Limitations.....	31
4 Specialized Services.....	33
4.1 Database Access Service.....	34
4.2 VIPAR Service for Bioinformatics Resources.....	37
4.2.1 Architecture.....	37
4.2.2 RDF Ontology.....	39
4.3 Conclusions.....	42
5 Framework Evaluation.....	43
5.1 Comparison Metrics.....	43
5.1.1 Metric Definition and Scope.....	48
5.1.2 Survey of Workflow Tools.....	51
5.2 WebSphere Application Development Integration Edition.....	52
5.2.1 Comparison Matrix.....	53
5.3 i-Flow Workflows.....	55
5.3.1 Comparison Matrix.....	55
5.4 TurboBench.....	58
5.4.1 Comparison Matrix.....	60
5.5 Visual Integrated Bioinformatics Environment (VIBE).....	62
5.5.1 Comparison Matrix.....	63
5.6 BINGO.....	65
5.6.1 Comparison Matrix.....	66

5.7 Workflow based Architecture to support Scientific Applications (WASA)....	68
5.7.1 Comparison Matrix.....	69
5.8 GeneFlow.....	71
5.8.1 Comparison Matrix.....	73
5.9 Bioinformatics Workflow and Data Management System (GeneBeans).....	74
5.9.1 Comparison Matrix.....	75
5.10 Conclusions.....	76
6 Conclusions.....	78
6.1 Summary.....	78
6.2 Future Developments.....	79
6.3 Latest in Technology and Possible Research Areas.....	81
6.3.1 Open Grid Service Architecture.....	81
6.3.2 Web Services Flow Language (WSFL).....	82
6.3.3 Web Services Conversation Language (WSCL).....	83
6.3.4 BizTalk.....	84
6.3.5 Open Source Workflow Projects.....	85
6.3.6 Bio* Projects.....	86
6.3.7 Discovery Link – IBM’s Life Sciences Framework.....	87
References.....	88
Appendix A: List of Tools Surveyed.....	94



## LIST OF TABLES

Table 2.1. WSDL Description.....	11
Table 3.1. Performance Measure.....	30
Table 4.1. RDF Description.....	40
Table 5.1. SDMSWS Vs WSAD IE.....	53
Table 5.2. SDMSWS Vs i-Flow.....	55
Table 5.3. SDMSWS Vs TurboBench.....	60
Table 5.4. SDMSWS Vs VIBE.....	63
Table 5.5. SDMSWS Vs BINGO.....	66
Table 5.6. SDMSWS Vs WASA.....	69
Table 5.7. SDMSWS Vs GeneFlow.....	73
Table 5.8. SDMSWS Vs GeneBeans.....	75

## LIST OF FIGURES

Figure 2.1. Web Service Architecture.....	7
Figure 2.2. Web Service Stack.....	8
Figure 2.3. Pipeline and Workflow.....	15
Figure 2.4. Web Service-Workflow Architecture.....	16
Figure 3.1. SDMSWS Architecture.....	19
Figure 3.2. UDDI Setup.....	21
Figure 3.3. SDMSWE.....	24
Figure 3.4. Bioinformatics Workflow Service Architecture.....	26
Figure 3.5. Gene-Expression Workflow.....	28
Figure 4.1. Extending Bioinformatics Environment.....	34
Figure 4.2. Database Access Service.....	36
Figure 4.3. VIPAR Architecture.....	38
Figure 5.1. Problem Solving Environment Layering.....	45
Figure 5.2. i-Flow Workflow Environment.....	57
Figure 5.3. TurboBench Workflow Environment.....	59
Figure 5.4. BINGO Workflow Environment.....	65
Figure 5.5. GeneFlow Workflow Environment.....	72
Figure 6.1. Flow Model.....	82

# 1 Introduction

A Problem Solving Environment (PSE) is a computer-based system that provides all the computational facilities necessary to solve a target class of problems efficiently, that is, it efficiently supports a specific set of scientific workflows [2]. A special class of PSE's are those that rely on networks. Network-based PSEs are collections of distributed applications, interfaces, libraries, databases, programs, tools, clients, and intelligent agents, which facilitate user interaction and cooperative execution of the components charged with the solution tasks [11]. Thus, the need for effective and efficient communication among these PSE components is obvious. This has resulted in a proliferation of communication building blocks, or middleware, for distributed scientific computing and problem solving [3]. The most recent, and quite promising, option is the availability of network-based services, sometimes called Web Services.

## 1.1 Issues

Development of enabling PSEs where scientists may share, integrate and utilize solutions directly in their research to enhance collaborative problem solving capabilities tends to be complicated, among other things, by:

- Use of Complex Data Structures: For example, situations where the applications are designed and programmed with specialized complex data structures that may not be easy to access from other applications.

- **Distributed Data Resources:** Another situation arises when the data resources are scattered around the network both physically and through the applications that access them. If, in such cases, there is also a lack of means to bridge distance and application interface gaps, we may experience inter-operability problems.
- **Application Customization:** Algorithms and solutions are often customized to the needs of specific research communities making it difficult for others to adapt to it or benefit from it.
- **Absence of Process Automation:** Manual steps, represented by the human intervention in a process, can be quite a production bottleneck. In a research environment it is highly desirable to automate certain steps of a scientific process.
- **Network Inaccessible:** Inability to access network-based applications and possible lack of standards for underlying communication protocols.

The classical construction approach to PSE's, as demonstrated in prior work has revolved around many different issues. Some of them are (1) Use of distributed application modules, where each module performs tasks and responds to messages from other participants in the system. In such an architecture, failure of one module can affect the performance of the whole system [4]; (2) Development of hand-crafted or home-grown solutions which may not function when incorporated into another PSE [5]; (3) Inability to support integration of a wide range of computational tools from different domains without requiring end-users to know a lot about the details of the underlying system and its architecture [6].

The need to support standardized communication among distributed applications has resulted in a proliferation of standard communication methods. Many were applied to distributed scientific computing. The latest approach is through a new breed of network-based services, called Web Services. Based on standardized communication protocols such as UDDI [7], SOAP [8], WSDL [9] and XML [10], Web Services have made it possible for organizations to communicate and integrate network-based services. Applications from disparate platforms can exchange data and information at both the network level and the web interface level. The idea is to enable services from different organizations and domains to interoperate seamlessly irrespective of language, platform and internal protocols. In scientific community, rapid advancement in both large-scale data acquisition and distribution, and relevant modular analysis techniques, supports the move towards service-oriented framework. However, the feasibility of use of service-based solutions in complex scientific environments is still an open issue.

## 1.2 Contribution

This work is concerned with evaluation of the feasibility, extensibility, usability and effectiveness of service-based support for scientific workflows. It has two components:

**Framework.** A successful open source proof of concept architecture and framework was developed using a Bioinformatics scientific problem solving workflow. Access to the data, computations, and user interfaces is based on the services architecture and standards such as XML for data descriptions, WSDL for service descriptions, SOAP for service delivery, and UDDI for service registration and brokering. The developed framework,

called Scientific Data Management Service Workflow System, or SDMSWS, was found to be sufficiently flexible and versatile that it was possible to effect appliance-like workflow composition and execution. An open-source solution, such as SDMSWS, enables services from different organizations, platforms and domains to interoperate seamlessly through standard interoperability protocols. This is contrasted with similar systems that implement proprietary service solutions.

**Assessment.** In order to compare SDMSWS with other systems that support scientific problem solving workflows, it was necessary to develop a set of measures. They include measures related to end-user issues, workflow issues, services issues, networking issues, and a variety of more general issues. The selection and the definition of the metrics, and the rationale behind them are discussed. A comparative analysis of SDMSWS with some standard commercial and research based scientific workflow tools reveals that its open source nature has some distinct advantages. For example, the framework minimizes the customization and integration of new component workflow services, and it shows that, unlike commercial solutions, it is relatively independent of the service domains and can incorporate any network-based service that conforms to standard Web Services description and protocols. It is the finding of this work that complex distributed scientific workflows can be, and should be, supported using open-source service-based solutions. However, current standards for workflow description, interchange and execution were found wanting and further work would be needed before one can depend on them in practical scientific workflow environments.

In the Problem Solving Environments, solutions proposed for complex processes and workflows from distributed resources comprised of single host, multiple hosts or network-based but none of them supported a service-based approach. To the best of the author's knowledge, this research is the first attempt that focuses on examining the feasibility, viability and usability of such service based support of scientific workflows and an assessment by problem solving environment based comparative analysis of the same with some standard scientific workflow tools.

### **1.3 Organization of Thesis**

Chapter 2 defines web services and scientific workflow technology. It describes concepts of relevant technologies namely, UDDI, WSDL, SOAP and workflows. Chapter 3 describes the architecture, functionality and components of the SDMSWS framework. A specific case study is presented for composition and integration of bioinformatics data acquisition and analysis services into an end-user workflow. Chapter 4 defines certain domain specific specialized services that can be integrated with SDMSWS to form an extended bioinformatics environment with emphasis on the databases, and tools, and agent based software used for specialized scientific investigation. Chapter 5 presents a comparative evaluation of the proposed architecture with respect to existing commercial and research based technologies based on similar principles. Chapter 6 concludes the thesis with a summary of the achievements, suggestions on possible future work, and a review of the current state of Web Service, workflow and bioinformatics research.

## **2 Web Service and Workflow Concepts**

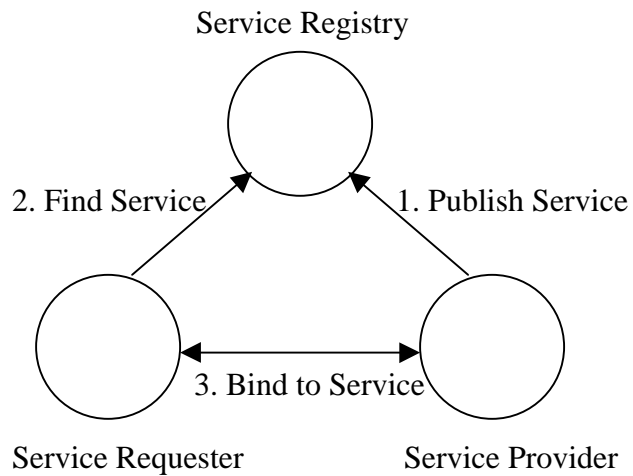
### **2.1 Web Service**

Web Services are software components that are self-containing, self-describing modular applications that can be published, located, and invoked across the Web. They allow applications to interoperate in a loosely coupled environment, discovering, and connecting dynamically to services without any previous agreements having been established between them. More importantly, a Web Service may combine several applications that a user needs. For the end-user, however, the entire infrastructure will appear as a single application [12]. Web Services encompass just about any application available over or delivered via the Web using standard protocols like Simple Object Access Protocol (SOAP), Universal Description, Discovery, and Integration (UDDI), Web Services Description Language (WSDL) and Extensible Markup Language (XML). The Web Service itself is really nothing more than a software program; say a java servlet or a java server page. The Web Services architecture is simply a wrapper for accessing this pre-existing code in a platform and language-independent manner. These technologies provide a standard means of communication among different software applications involved in presenting dynamic context-driven information to the user.

#### **2.1.1 Architecture**

There are three participants (Figure 2.1) in the Web Services architecture [1]:





**Figure 2.1.** Web Service Architecture

- Service Provider: Provide services, and maintain a registry that makes those services available.
- Service Registry: Clearinghouses for services. Service registry acts as a central location for registering all services.
- Service Requestor: Looks for Web Services of use, and then invokes those services.

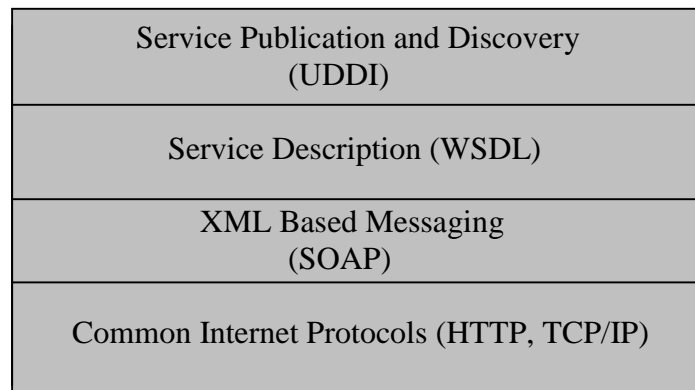
### 2.1.2 Operations

There are three Web Services operations [13]:

- Publish/Unpublish: Publishing and unpublishing involves advertising services to a registry (publishing) or removing those entries (unpublishing). The service provider contacts the service broker to publish or unpublish a service.

- Find: Service requestors and service brokers perform the find operation together. The service requestors describe the kinds of services they are looking for, and the service brokers deliver the results of such requests.
- Bind: The bind operation takes place between the service requestor and the service provider. The two parties interact and then the requestor accesses and invokes services of the provider.

## 2.2 Web Service Stack



**Figure 2.2.** Web Service Stack

The Web Service Stack defines the Web Services architecture layering [13]. (Bottom-Up)

- Layer1: Standard Internet protocols like the HTTP and TCP/IP.
- Layer2: Simple Object Access Protocol (SOAP) is based on XML and is used for exchange of information between services from organizations.
- Layer3: Web Services Definition Language (WSDL) is used for describing service attributes.

- Layer4: Universal Description, Discovery and Integration, which is a central repository for holding and describing services.

## **2.3 Universal Discovery, Description and Integration (UDDI)**

The UDDI (Universal Description, Discovery and Integration) registry is a global, public, online directory that gives organizations a uniform way to describe their services, discover other organizations services, and understand the methods necessary to conduct information exchange with one another. UDDI's approach to discovery is to have a registry of services distributed across the Web. In that distributed registry, businesses and services are described in a common XML format. The structured data in those XML documents is easily searched, analyzed, and manipulated. The term "dynamic discovery" is used in a UDDI context, as the common use case for UDDI involves a client application dynamically finding an exchange partner via a UDDI registry and then deciding to exchange information [8]. Publishing/Describing Web Services implies service users do not have to “re-invent the wheel” every time they want to do something that others have already done.

The UDDI information model contains four core elements [14]:

- Business information: This is described using the business entity element, which represents a physical organization. It contains information (such as name, description, and contacts) about the organization. The business entity information

includes support for “Yellow Pages” type searches that can be performed to locate organizations that service a particular industry or category.

- Service information: This is described using the business service element, which groups together related services offered by an organization.
- Binding information: This is described using the binding template element, for information that is relevant for application programs that need to connect to and then communicate with a remote Web Service. The instructions may either be in the form of an interface definition language such as WSDL, or a text-based document.
- Information about specifications for services: This is described using the tModel element, which is an abstract representation of the technical specification. A tModel has a name, publishing organization, and URL pointers to the actual specifications themselves.

## **2.4 Web Service Description Language (WSDL)**

WSDL (Web Service Description Language) defines the XML grammar for describing services as collections of communication endpoints, or ports, capable of exchanging messages irrespective of the underlying network protocols. It is used to describe what a Web Service can do, where it resides, and how to invoke it. UDDI registries describe numerous aspects of Web Services, including the binding details of the service. WSDL fits into the subset of a UDDI service description. Companies can publish WSDL's for

services they provide and others can access those services using the information in the WSDL. Table 2.1 is a sample WSDL description:

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://152.1.158.185:9080.....
.....XMLSchema">
  <wsdl:message name="getDataResponse">
    <wsdl:part name="return" type="xsd:string" />
  </wsdl:message>

  <wsdl:message name="getDataRequest">
    <wsdl:part name="input" type="xsd:string" />
  </wsdl:message>

  <wsdl:portType name="DataService">
    <wsdl:operation name="getData" parameterOrder="input">
      <wsdl:input message="intf:getDataRequest" name="getDataRequest" />
      <wsdl:output message="intf:getDataResponse" name="getDataResponse" />
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="urn:sdsc.service.BlastOneSoapBinding" type="intf:DataService">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="getData">
      <wsdlsoap:operation soapAction="" />
      <wsdl:input name="getDataRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://152.1.158.185:9080/axis/services/urn:sdsc.service.
          BlastOne/axis/services/urn:sdsc.service.BlastOne" use="encoded" />
      </wsdl:input>
      <wsdl:output name="getDataResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://152.1.158.185:9080/axis/services/urn:sdsc.service.
          BlastOne/axis/services/urn:sdsc.service.BlastOne" use="encoded" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="DataServiceService">
    <wsdl:port binding="intf: urn:sdsc.service.BlastOneSoapBinding"
      name="urn:sdsc.service.BlastOne">
      <wsdlsoap:address
        location="http://152.1.158.185:9080/axis/services/urn:sdsc.service.BlastOne" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

**Table 2.1.** WSDL Description

WSDL description (Table 2.1) uses the following elements to define Web Services [9]:

- Types: Containers for data type definitions using some type system (like XSD).
- Message: An abstract, typed definition of the data being communicated.
- Operation: An abstract description of an action supported by the service which could be one of the following four types, one way, request-response, solicit-response and notification.
- Port Type: An abstract set of operations supported by one or more endpoints.
- Binding: A concrete protocol and data format specification for a particular port type.
- Port: A single endpoint defined as a combination of a binding and a network address.
- Service: A group of related ports.

A service implemented by a Java class could define one or more operations. Operations would map to individual Java methods, implemented by the class. Messages would map to the return types of the methods, or to their parameter list. WSDL also defines a common binding mechanism. This is used to attach a specific protocol or data format or structure to an abstract message, operation, or endpoint. This allows the reuse of abstract definitions. WSDL provides specific binding extensions to SOAP, HTTP GET/POST and MIME formats and protocols, but any other binding mechanisms can also be used [9].

## **2.5 Simple Object Access Protocol (SOAP)**

SOAP, which is built on XML, defines a simple way to package information for exchange across system boundaries, allowing one program to invoke service interfaces across the Internet, without the need to share a common programming language or distributed object infrastructure. SOAP contains the framework where message composition and their responses can be specified. This protocol is specific mostly to Web Services over HTTP. A Web Service could interact with remote machines through HTTP's post and get methods, but SOAP is much more robust and flexible.

### **2.5.1 SOAP Message**

A SOAP message is an XML document that consists of a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body [7].

- The SOAP envelope construct: Defines an overall framework for expressing what the message contains, who should deal with it, and whether it is optional or mandatory.
- The SOAP encoding rules: Defines a serialization mechanism that can be used to exchange instances of application-defined datatypes.
- The SOAP RPC representation: Defines a convention that can be used to represent remote procedure calls and responses.

## 2.6 Scientific Workflows

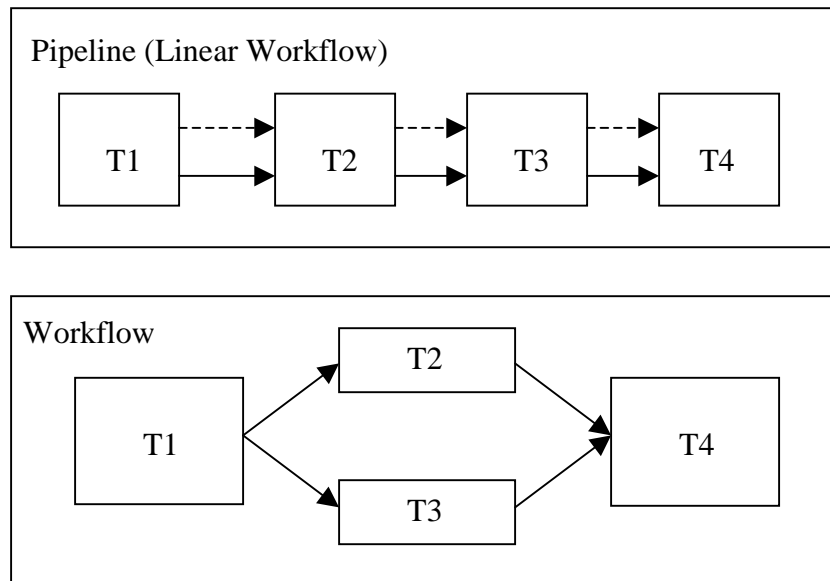
In many sciences and engineering areas computation is complex and repetitive. Graph based notations, like the generalized activity networks (GAN) are a natural way of representing numerical and human processing [57]. Workflows are similar representations of structured activities [11]. A scientific process usually involves the invocation of a series of activities that are typically invoked in a routine manner.

Workflow computations involve the following issues:

- Formatting output from one task, feeding it as input to the next task and adding additional parameters if necessary.
- As scientific data sets are consumed and generated by simulation programs, the intermediate results are checked for consistency and validated to ensure that the computation as a whole remains on track.
- Scientific computations can involve much human intervention. Roles of the participants involved must be explicitly represented to enable effective intervention.
- Successful execution of workflows requires careful handling of exceptions during definition and execution of tasks.
- Workflows should be controlled from a single point, giving user the capability to perform computations that span across distributed resources.



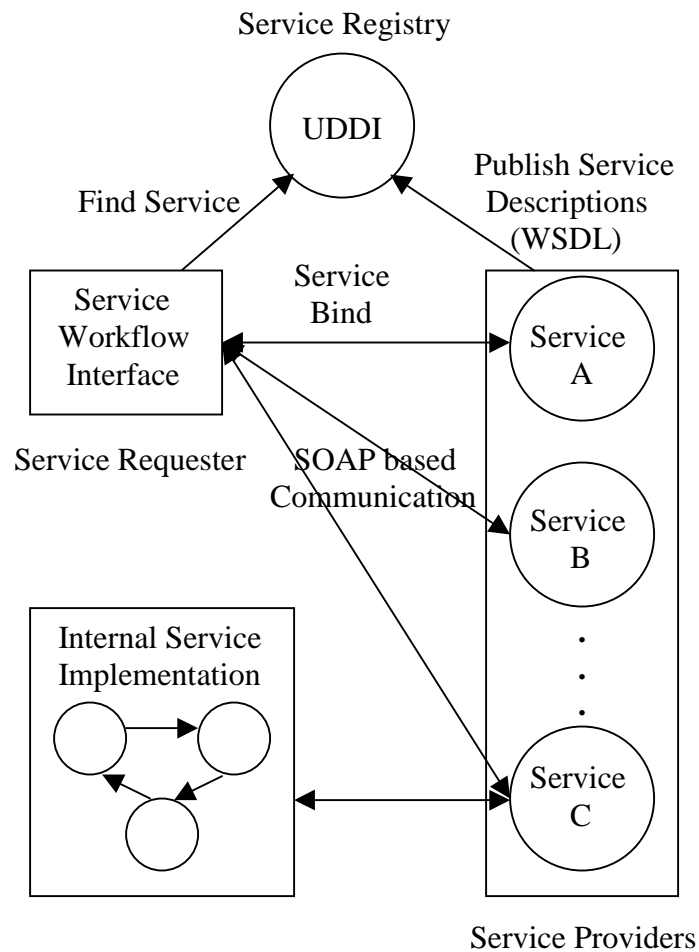
### 2.6.1 Pipeline and Workflow



**Figure 2.3.** Pipeline and Workflow

Figure 2.3 depicts a pipeline (linear workflow) and a workflow, solid arrowed lines represent flow of processing control and dotted lines indicate flow of information between activities. Workflow and pipeline represent series of tasks to be completed according to some sequence. A task (T) in its most basic definition is the minimal unit of work; it can have inputs and outputs. Tasks are linked together by connectors, which represent joins or forks. Pipeline is a simplified workflow; it is linear. A pipeline has a single input and a single output. In a pipeline the sequence of tasks is executed linearly. A workflow can be made up of multiple linear flows and can have multiple outputs but mostly has a single point of entry. It can consist of multiple paths from the start to the end task with branches in between. Workflows can also have decision nodes, AND nodes and OR nodes.

## 2.7 Web Service-Workflow Architecture



**Figure 2.4.** Web Service-Workflow Architecture

Figure 2.4 describes the architecture integrating Web Services and workflows. Service providers register service and service descriptions (WSDL) with the public registry (UDDI). Service requestor composes a workflow consisting of services. It interfaces only with the public descriptions of the service and not their internal implementation. Communication between the service requestor and the service provider is supported by SOAP over standard network protocols.

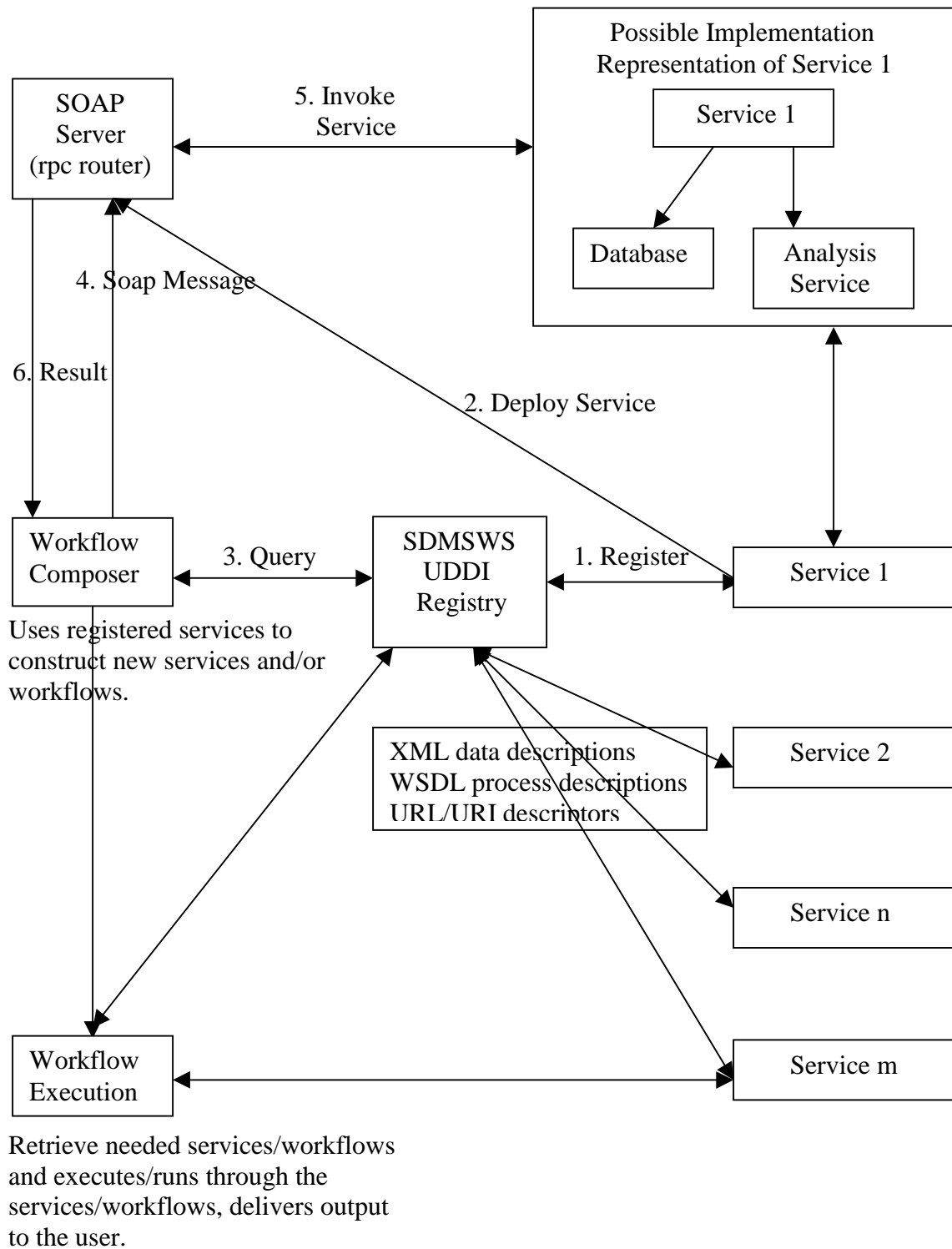
Designing and implementing complex workflows often requires interdisciplinary collaborations involving aspects of scientific computing and data integration. Abstraction and encapsulation are standard computer science techniques for shielding end-users from complexities of scientific computing. In collaborative and distributed systems additional complexities arise from heterogeneity of the distributed environment. The following chapter addresses this problem by implementing a service oriented scientific workflow framework that supports integration of network-based distributed services based on the standard Web Services interoperability protocols.

### **3 Scientific Data Management Service Workflow System (SDMSWS)**

The following is a pilot implementation of a service based scientific workflow framework that provides effective access to diverse, distributed scientific tools across different organizations using the de facto Web Services technologies (UDDI, SOAP and WSDL). The idea is that every organization should publish their service descriptions with a centralized registry and have full control on how services interact and execute. The interested user will be able to locate these services and invoke them either on their own or as parts in a workflow. The framework will expose the workflow to the outside world as a means of enabling services from different organization to interoperate seamlessly where services communicate with each through standard interoperability protocols and perform the tasks defined in the workflows. The specific case study is composition and integration of Bioinformatics data acquisition and analysis services into an end-user workflow.

#### **3.1 Architecture, Working and Components**

The Scientific Data Management Service Workflow System (SDMSWS) framework is proposed with the idea to develop, implement, test and deploy an integrated service-based system for support of scientific workflows. It assists in composing and executing workflows from services using the workflow construction and execution editor and making the access to scientific data and tools as devoid of unnecessary technological overhead as possible. Figure 3.1 represents the overall architecture of the system.



**Figure 3.1.** SDMSWS Architecture

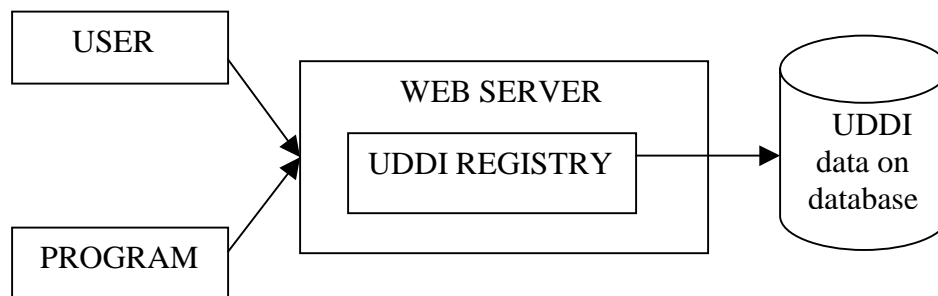
Every Organization exposes and registers its services, which may have been developed on different platforms, with the UDDI registry. The service provider has to publish specifications, and the messages it can exchange, about the service namely access point (where its been hosted and deployed), methods implemented, input/output parameters and other service related information. This can be a description in natural language, an XML document or in Web Services Definition Language. The service is deployed to a SOAP router (Axis [15], which is Apaches implementation of SOAP). The SOAP router is a Remote Procedure Call (RPC) router that accepts and invokes client requests. The client can query the UDDI to find services of interest and use the workflow tool to construct and execute workflows from the services available. The client need not know the implementation details of a service. The workflow tool accepts parameters, from the service description file, namely service name, method name, input method parameters and SOAP URL, and creates a SOAP call that is sent to the SOAP router. The SOAP router calls the appropriate service based on the parameters it receives in the SOAP message. Finally, it returns the result to the client.

The SDMSWS has the following components:

- Universal Description Discovery and Integration: UDDI registry serves as the central registry for services and is accessible either directly or through programs. It consists of a UDDI registry, registry database for storing information and a web server as a servlet engine that also provides security features like user authentication. Both human and program can access the UDDI registry. User can

browse the UDDI web page to find and use a service of interest. Figure 3.2 depicts the setup of the UDDI. It consists of:

1. UDDI Registry: IBM UDDI Registry implementation. (part of IBM WSTK 2.3 [16]).
2. Registry Database: For storing the registry information (on uddidata database).
3. Apache Tomcat 4.0 [17] Web Server: Tomcat, which belongs to the class of Web Servers from Apache Software Foundation, is a servlet container that is used in the official reference implementation for the java servlets and java server pages. [A web server serves web pages to clients across the Internet. The web server hosts pages, scripts, and programs and serves them using HTTP].



**Figure 3.2.** UDDI Setup

- Service: A service is a program that can receive requests and respond to them. The service may be presented in the form of a program like a servlet running on a web server anywhere on the network. The service details namely owner, access point and contact information are submitted to the UDDI. The methods

implemented, input-output parameters and other details of the service are published in the WSDL description corresponding to that service. Following are the components of a service [18]:

1. Service Interface: A representation of the service to the outside world and the available operations and messages that can be exchanged.
  2. Service Binding: Binding provides the connection between the service interface and the implementation and information on how to get to it.
  3. Service Implementation: This is the implementation of the service. (SOAP, EJB, Bean etc). Service implementation has two parts,
    - Service Provider: Offers some function or process as a service.
    - Service Consumer: Requests the service to perform some task.
- SOAP Module: The SOAP module is an integral part of the framework:
    1. A Service provider develops and deploys services on the SOAP router. The services are registered with the UDDI Registry.
    2. The developer of the workflow queries the UDDI registry to enquire the service description details.
    3. SDMSWS Workflow Editor (SOAP client) let's the end users construct and execute workflows using the service description attributes.
    4. When the workflow is invoked, it makes an RPC to the SOAP Router through appropriate SOAP messages. In the SOAP message, it includes information about the parameters required to invoke the service.



5. The SOAP server on receiving the SOAP message invokes the actual implementation of the service either over the network or locally depending on the location of the service. It returns the result of the service invocation to the end user.
  6. Note [AXIS as SOAP Server]: “Axis is essentially a SOAP engine. A framework for constructing SOAP processors such as clients, servers, gateways, etc. It also includes a stand-alone server that can plug into servlet engines like Tomcat” [15].
- WSDL and XML Process Descriptors: WSDL is used to describe the specification of a service and express the information about the interface and its bindings. There is information for automatic invocation of a service by another service without human intervention at a much lower level. An example WSDL description is described in chapter 2.
  - Scientific Data Management Services Workflow Editor (SDMSWE) [The Editor was developed by Ms. Sangeeta, with contributions from Mr. Cheng and myself]: The workflow tool lets the user define, construct and execute the sequence of data and analysis services it wants to invoke. It creates a Web Service based environment for executing and monitoring workflows. The workflow editor has the following properties:
    1. GUI based interface used to create, edit and save linear workflows that involve multiple services and data resources. (Figure 3.3)

2. The workflow consists of task nodes that are connected into a directed graph. Each task represents a unit of work to be done. It may be invoking a service on the remote machine, or a service over the web, or even a local program. It may also just deliver information from a web page.
3. In the workflow editor, the arrows (originating from the START node) define the sequence and ordering of tasks and eventually the flow of data from one task to another.
4. The workflow editor has features for looping to iterate around one or more tasks.

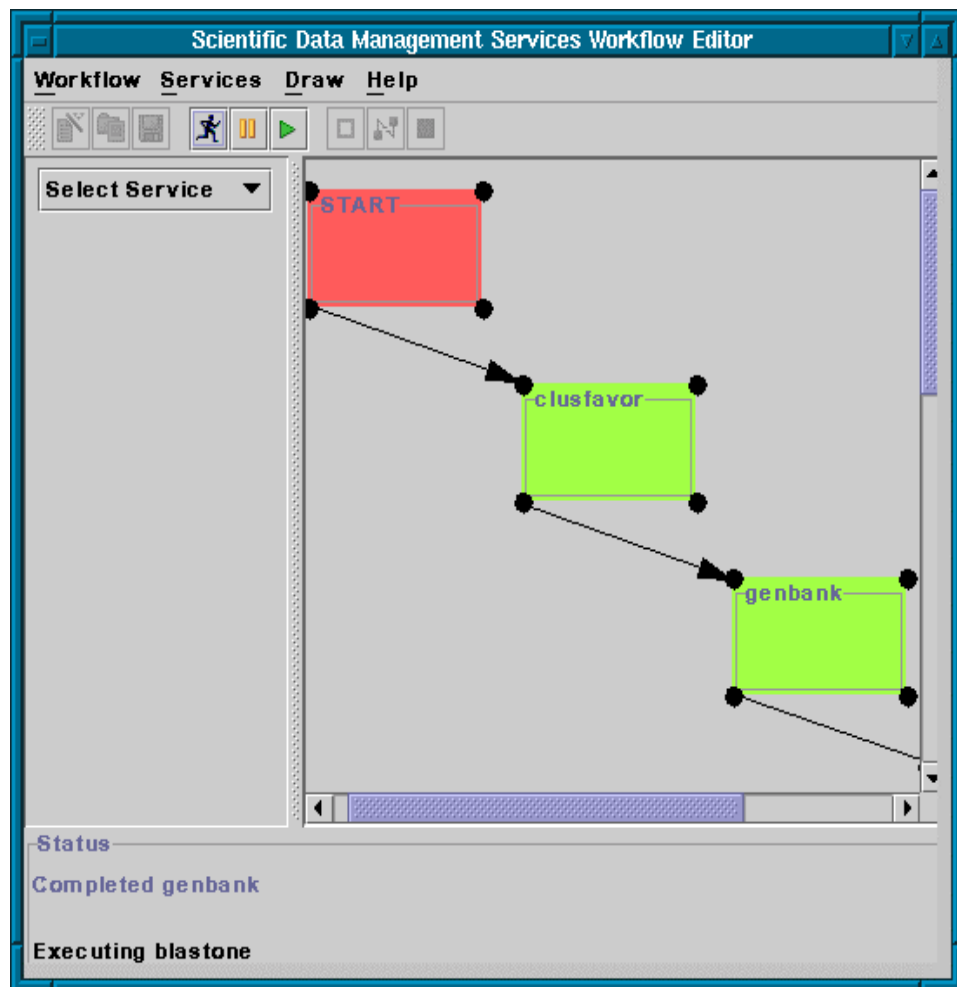


Figure 3.3. SDMSWE

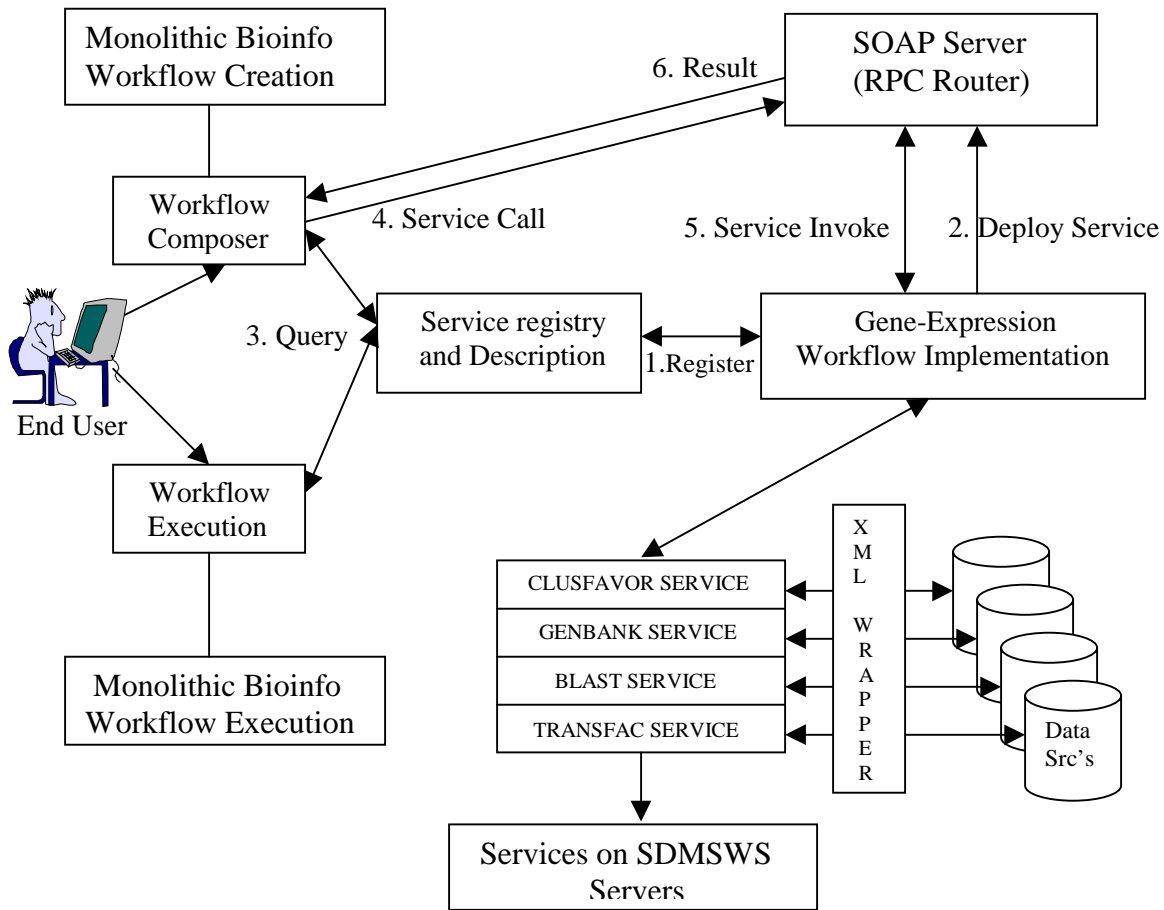
5. The workflow engine is central to the workflow editor architecture and is responsible for composing SOAP calls based on the service interface parameters supplied to the workflow components.
6. The workflow engine provides basic syntactical verification of workflow parameters. A more extensive verification would involve verifying the workflow description against the service description in the registry.
7. The workflow editor also provides for automation or manual intervention during the execution of a workflow.
8. The workflow editor can dynamically locate services registered with the UDDI and extract service details.

There are two modes of operation for the workflow GUI. One is the workflow composer mode used to construct workflows. The other is the playback mode where the end-user initiates and tracks the execution of a defined workflow. The execution engine invokes the services defined in the workflow. Alternatively the user can directly go to the UDDI registry, find the interested service, and invoke those services directly (as a web page).

## **3.2 Bioinformatics Workflow Service**

We have implemented a prototype of a bioinformatics workflow that chains together the following bioinformatics web resources Clusfavor [19], NCBI Genbank [20], NCBI Blast [21] and Transfac [22] in a workflow, thus automating a scientist's manual procedure of

interacting with each of these sites individually. Each of these services performs certain analysis on the data that is provided to it. This monolithic bioinformatics workflow services run on the SDMSWS servers as Gene-expression workflow implementation. Registered with UDDI, they are invoked through SDMSWS workflow GUI.



**Figure 3.4.** Bioinformatics Workflow Service Architecture

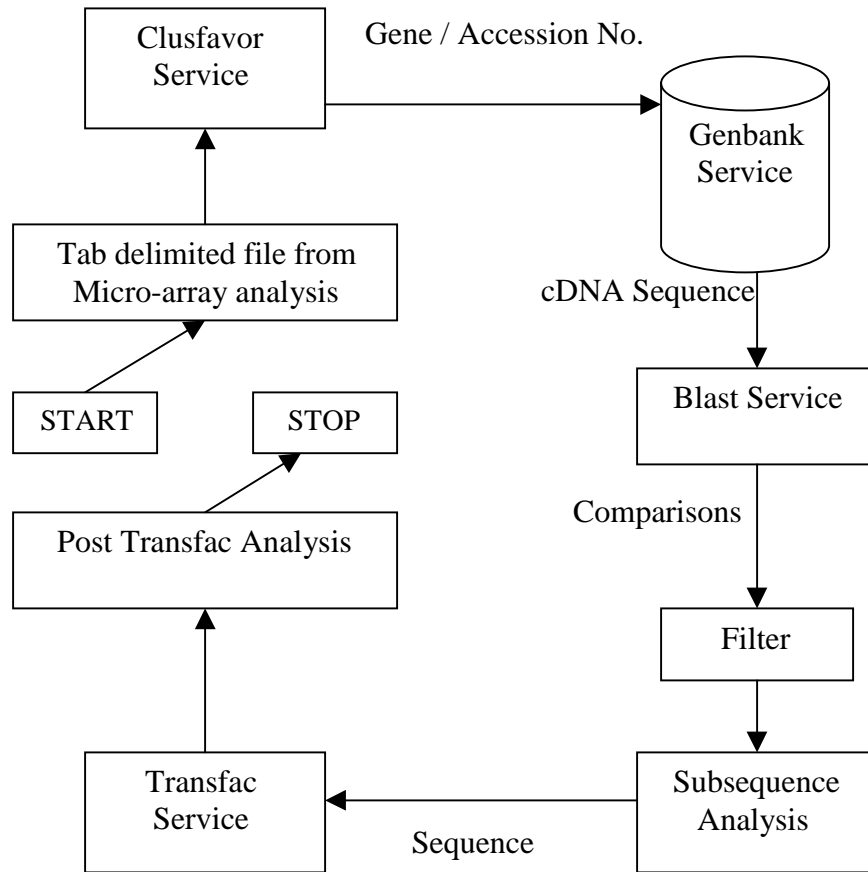
In figure 3.4 the workflow editor is used to compose a scientific workflow and associate each task with one of the following services, Clusfavor, NCBI Genbank, Blast and Transfac. Input parameters needed to invoke the services are obtained from the WSDL description at the UDDI. Once the tasks receive the parameters, the SOAP client

(Workflow GUI) sends an invocation to the SOAP server. The SOAP server contacts the Gene-Expression workflow implementation, which is the implementation of these services and invokes the actual data sources. The service is executed and the result is returned to the SOAP server, which in turn returns the result to the end user. A wrapper [23] interface is used to filter the result as XML data. The output of one task can either be passed directly or modified and passed as the input to the next task in sequence. All the outputs are stored at the client side and provide either a browser view or an XML view of data. A composed workflow can be saved and played again at a later stage thereby saving time in successive runs of an experiment.

### **3.2.1 Gene-Expression Workflow**

[This workflow is proprietary to Dr. Matthew Coleman of Lawrence Livermore National Labs.] The workflow has been divided into four basic steps, namely Clusfavor analysis, GenBank search, Blast search, and Transfac search using MatInspector tool [22]. A user enters a set of relevant parameters upfront. After this, the workflow is automatically executed, using the parameters along the way. Simple iterations of the same step can be executed for multiple inputs. Whenever there are multiple possible outcomes or choices from a step, a certain decision can be made about the parameters that would be passed as input to the next task in sequence. We have identified and implemented this bioinformatics workflow service that is now being used as the bootstrapping research and development case study. It involves access to multiple network-based data-sources and other annotation databases (access via XML), as well as access to other information

resources. Figure 3.5 demonstrates the input and output parameters, and the sequence of execution, of the components in the Gene-Expression workflow.



**Figure 3.5.** Gene-Expression Workflow

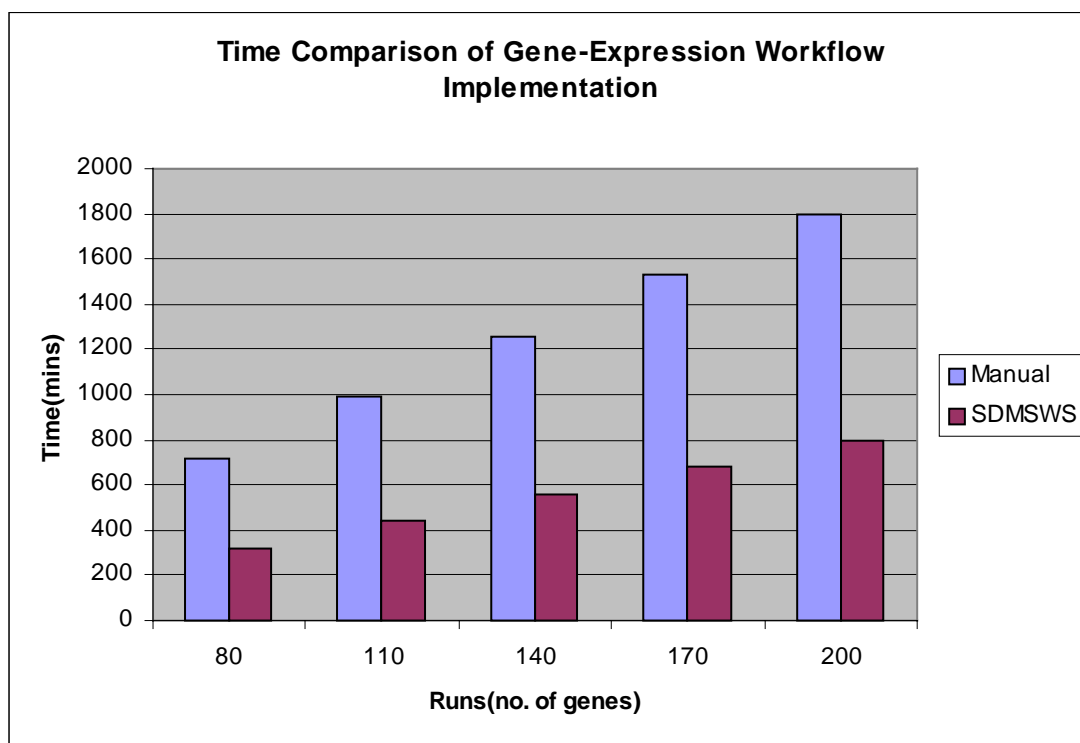
In Gene-Expression workflow (Figure 3.5) we feed the data of a given microarray analysis to a cluster analysis tool (Clusfavor). Based on the results select the gene identifiers and retrieve their cDNA sequences from the GenBank database. We then perform a sequence comparison (Blast) of the retrieved cDNA sequences against known sequences and analyze results using Transfac with defined parameters. Following is a brief description of the components in the workflow:

- Clusfavor (Cluster and Factor Analysis with Varimax Orthogonal Rotation): A Windows based computer program for principal component and unsupervised hierarchical cluster analysis of DNA microarray transcriptional profiles [19]. It takes a tab-delimited file as an input, performs analysis, and outputs a group of genes. Clusfavor is used to obtain cDNA sequences for genes of interest. Output of Clusfavor consists of several clusters of named sequences that have similar expression characteristic. It is a collection of clustered cDNA fragments. Clusfavor service is encapsulated as a SOAP service and deployed.
- Genbank: Genbank is the NIH genetic sequence database, an annotated collection of all publicly available DNA sequences [20]. A cDNA string is extracted from the Clusfavor output and is passed to Genbank to do a lookup and obtain a sequence of cDNA strings represented by the cDNA string extracted from the previous output. This service uses a single accession number and makes a request to NCBI's GenBank nucleotide database and returns the URL link for the page, this URL has the information about the gene corresponding to the accession number. The service then uses the target URL attribute from the previous service call and returns the nucleotide sequence for that gene.
- BLAST (Basic Local Alignment Search Tool): BLAST is a set of similarity search programs designed to explore all of the available sequence databases [21]. The service performs blast using the sequence htgs database. We obtain the blast results, and for those results that have a good match, identify where in the returned sequence the match occurred and take that part of the sequence +/- 1000 base pairs.

- Transfac: Fast and sensitive service for detection of consensus matches in nucleotide sequence data. It builds a consensus model using location and consensus of transcription factor binding sites [22]. The sequence generated from the previous BLAST service is furnished to the Transfac service.

### 3.2.2 Performance Measure

Table 3.1 compares the time to perform the Gene-Expression workflow tasks manually as against using SDMSWS.



**Table 3.1.** Performance Measure

The time for single manual invocation of the workflow involves the time to find relevant databases and online tools, manually copy outputs from one tool to another, possibly



port-process the output and paste appropriate pieces into the next process. On an average this process takes around nine minutes. Similarly the time for executing the same workflow using SDMSWS, which has just one time set up of the workflow parameters, takes an average of four minutes. The workflow in both cases is run for varying number of genes.

We notice a significant decrease in time while using SDMSWS over manual operations. This factor is even more significant when the workflow is invoked with a large number of input data (genes). Suppose the workflow is invoked for 200 genes, this means that it would take about  $200 * 9$  (time for manual invocation) equals 1800 minutes to run all 200 genes through the workflow manually. Not to forget the human stress factor of running such long iterations. To perform 200 such iterations it would take humans about 6-7 days. Compared to this the service workflow tool can run 200 genes in  $200 * 4$  (time for tool based workflow) equals 800 minutes. This would finish well within a day and would not require any human intervention at all. Including human factors into the workflow execution makes the contribution of SDMSWS framework's workflow automation worthwhile.

### **3.3 Current Limitations**

The following are some limitations of the current architecture.

- The system currently implements a text based workflow description, which needs to be replaced by a standard workflow definition language.

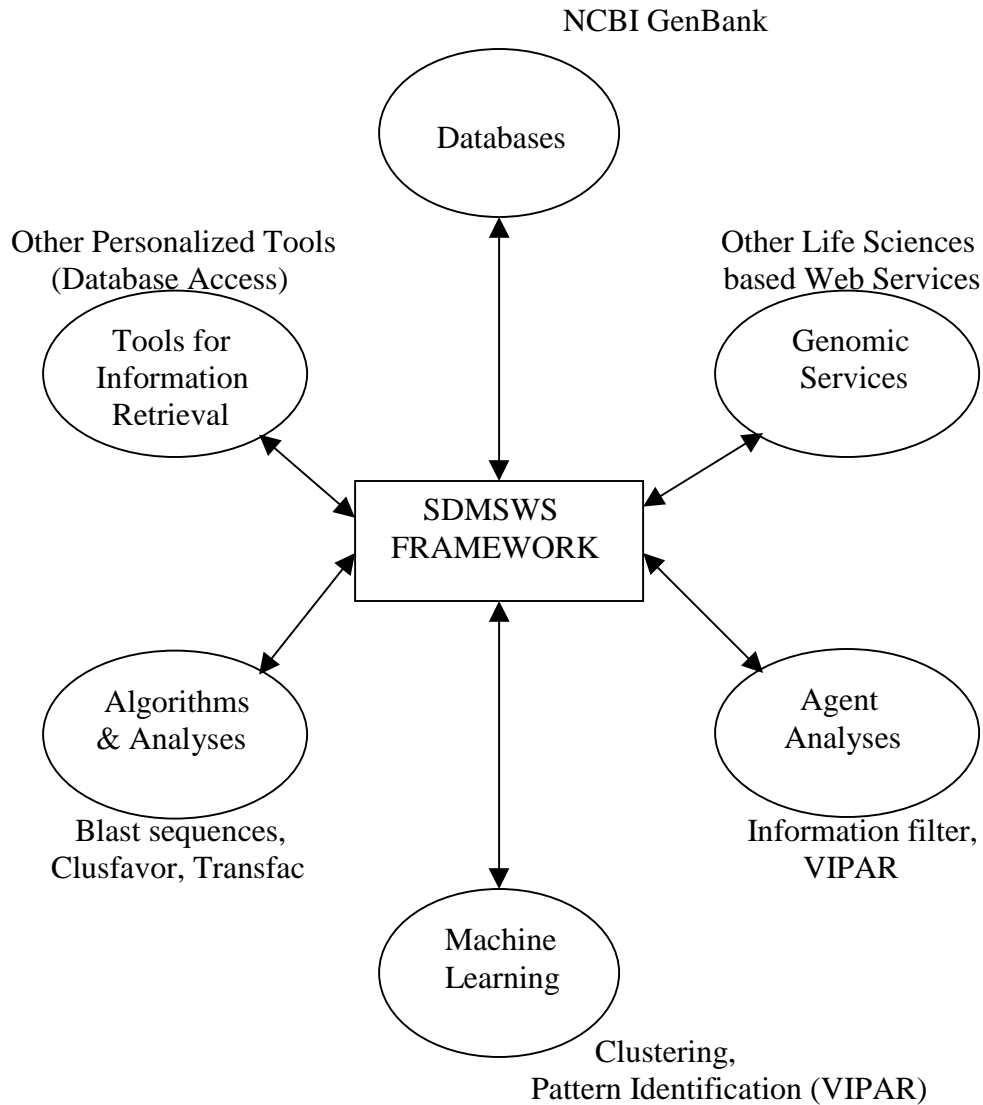
- Lack of workflow verification. We need to verify the services in the workflow with the service specification in the registry.
- Lacks support for branching within a workflow and parallel execution of workflow components.
- Very basic and primitive exception handling in the framework, which needs to be replaced by an extensive error control and recovery.
- Lack of browser based access to the workflow editor. Every time a newer version is released it has to be downloaded on the end user system.
- System uses the default web server security to guard services and registry. This needs to be replaced by a much more detailed security capability.

This chapter described the architecture and components of SDMSWS. In the later half we implemented a Bioinformatics domain specific service based workflow in support of the defined architecture. The following chapter talks about some other domain specific specialized services that can run in association with the SDMSWS and extend the framework to a much more diverse yet integrated Bioinformatics environment.

## 4 Specialized Services

Bioinformatics is the field of science in which biology, computer science, and information technology merge to form a single discipline. It is an emerging research discipline that uses information technology, mainly software tools, to organize and analyze scientific data in order to answer complex biological questions. It comprises of databases, algorithms, scientific and intelligent tools to collect, organize, store, retrieve and analyze biological data. Important disciplines within Bioinformatics and computational biology include the development and implementation of tools that enable efficient access, use and management of various types of information and analysis and interpretation of various types of data from different domains. “The ultimate goal in the field of bioinformatics is to enable the discovery of new biological insights as well as to create a global perspective from which unifying principles in biology can be discerned. This can be achieved only when there is effective scientific research that is spurred by intelligent data access and analysis across domains” [24].

A realistic environment would consist of a number of distributed domain specific data and information sources. SDMSWS framework can be central to such an environment providing flexibility, extensibility, and effective and efficient access to a consistent view of data from these sources through an intuitive and useful perspective. Following are some other domain specific specialized solutions that, along with the SDMSWS framework, are participants of this realistic environment. Some of them have been discussed in the previous chapter.



**Figure 4.1.** Extending Bioinformatics Environment

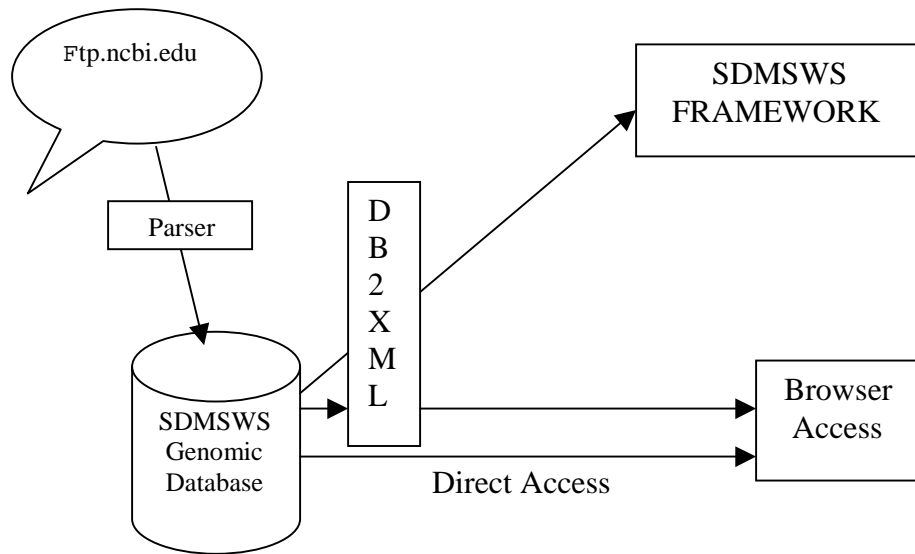
## 4.1 Database Access Service

For sometime now, a bioinformatics concern was the creation and maintenance of a database to store biological information, such as that of genes. Development of this type

of database involved not only design issues, but also the development of complex interfaces whereby researchers could both access existing data as well as submit new or revised data. Ultimately, however, all of this information must be combined to form a comprehensive picture of how researchers may study this data. Therefore, the field of bioinformatics has evolved such that the most pressing task now involves the analysis and interpretation of various types of data [24]. This plethora of information has led to an absolute requirement for computerized databases to store, and organize data, and for specialized tools to view and analyze the data.

A database is a large, organized source of persistent data, usually associated with computerized software designed to update, query, and retrieve components of the data stored within the system. A simple database might be a single file containing many records, each of which includes the same set of information. For example, a record associated with a genomic sequence database typically contains information such as gene name, gene id, the scientific name of the gene organism and often, literature citations associated with the sequence.

In figure 4.2 a parser accesses and filters data from NCBI Genbank repository to store it in the SDMSWS database. Data on the website is stored as files and thus data in the database represents the same information but provides better means of storage and querying. A service to retrieve data from the database is defined, registered and deployed with the SDMSWS framework. The data in the database can also be accessed from a web page.



**Figure 4.2.** Database Access Service

In figure 4.2 DB2XML [25] is a wrapper tool that transforms relational databases into XML documents. It provides attributes that describe characteristics of the data (meta data). It generates xml data as a result for a sql query to any database. DB2XML has two implementations; it can be implemented and invoked as a stand-alone tool from a graphical user interface or as a servlet to dynamically generate XML-documents. It works with any database provided the right database drivers are used. As XML is an accepted standard for data formatting, DB2XML wrapper could be used as an efficient way to define content dependent data.

## **4.2 VIPAR Service for Bioinformatics Resources**

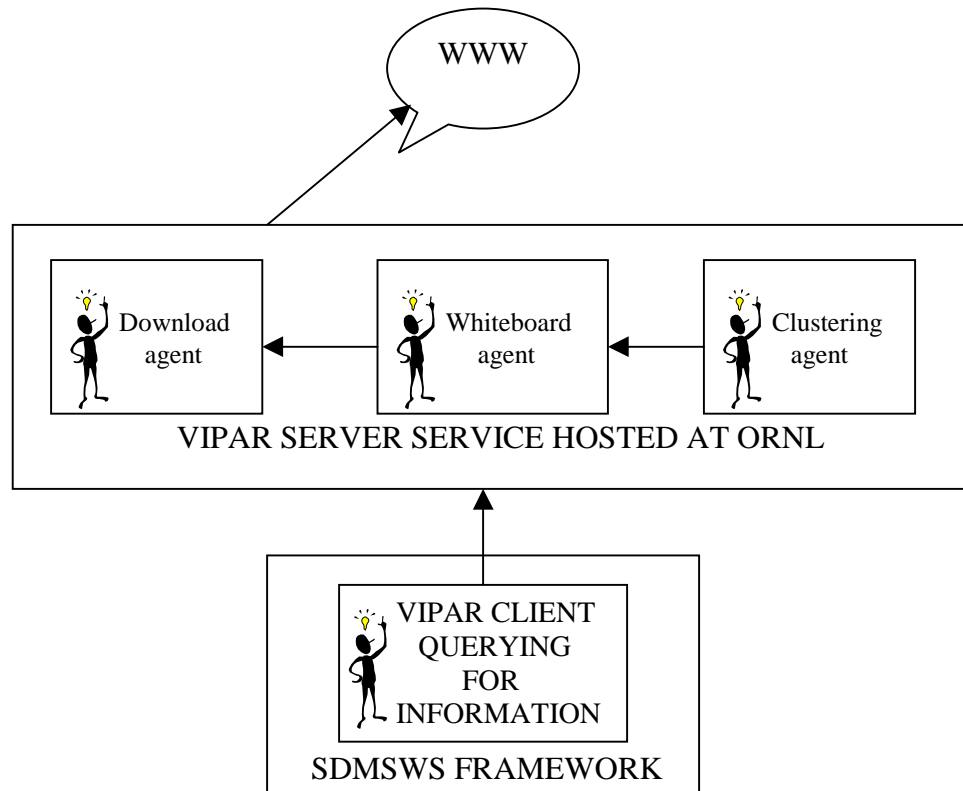
VIPAR (Virtual Information Processing Agent Research) [26] software developed by Oak Ridge National Labs (ORNL), demonstrates the ability to self-organize a number of related articles. It extends the field of information agents in a number of ways, most significant is the ability for agents to use a flexible Resource Description Framework (RDF) [27] ontology for managing information, and the ability to dynamically add and cluster new information entering the system [26]. To organize and classify information collected from various sources, VIPAR uses cooperative and adaptive information agents. These agents are able to work together to gather and organize information. There are different agent types that use a communication protocol that enables them to interact. For example, individual agents gather information (articles) from individual bioinformatics resources, some agents analyze the articles, and others organize the information. To deploy such agents VIPAR uses the Oak Ridge Mobile Agent Community (ORMAC) framework developed by ORNL. ORMAC is a generic agent framework providing transparent agent communication and mobility.

### **4.2.1 Architecture**

Virtual Information Processing Agent Research (VIPAR) is a client server application based on the RMI (Remote Method Invocation) technology. The server consists of three agents: -

- Download agent - Responsible for downloading all the information from the web.

- Whiteboard agent - Stores and manages the collected information. Other agents have to subscribe to this agent to access articles.
- Clustering agent - Performs clustering analysis on the collected data.



**Figure 4.3.** VIPAR Architecture

The server starts downloading information, from network based resources. These agents use the RDF ontology to capture and download information. Once downloaded these articles are posted on the whiteboard where they are stored, managed and checked for duplicity. The cluster agent then runs the algorithm on these articles clustering them on the basis of certain information provided to the agent. VIPAR client initiates a connection to the server and generates the graphical representation of the clustered information. The



VIPAR server can be queried for genomic information from the SDMSWS framework. The server runs as a service at Oak Ridge National Labs and is registered with the SDMSWS UDDI registry. The workflow editor can dynamically locate the VIPAR servers registered with the registry and can compose and invoke a query.

#### **4.2.2 RDF Ontology**

XML is an accepted standard to address the problem of organizing and classifying large amounts of heterogeneous information accessible over the Internet. However, a significant challenge is to automatically convert information currently expressed in a standard HTML format to an XML format [28]. Converting HTML information to XML is difficult as there is no single, uniform structure existing across most of the Internet resources, which would allow a simple conversion from the HTML format to the desired XML format. Even within a single web site the structure may not be consistent. The main issue is defining a common description that allows disparate HTML pages to be converted to XML in a consistent way. One means of performing this conversion is to use a generic parsing engine that can be driven by an ontology. The semantic web is one such approach that extends the current Internet by giving well-defined meaning to information [29]. The semantic web is based on the Resource Description Framework (RDF). RDF is a mechanism for using XML to describe an Internet resource as a directed graph [28].

Table 4.1 is an example of an RDF file that defines the model for extracting information from a source. There are five basic elements of an RDF description [28]:

```

<articleSourceName>
    Genomics Pharma
</articleSourceName>

<rootURLStr>
    http://genomics.phrma.org/
</rootURLStr>

<collection>
    Genomics
</collection>

<searchDepthLimit>
    2
</searchDepthLimit>

<minutesWaitBetweenDownloadSessions>
    60
</minutesWaitBetweenDownloadSessions>

<tocMetaData>
    <urlRegEx>
        http://genomics.phrma.org/today/
    </urlRegEx>
</tocMetaData>

<articleMetaData>
    <article>
        <urlRegEx>
            http://www\.\eurekaalert\.\org/pub_releases/[0-9]{4}-[0-9]{1,2}/.*\.\php
        </urlRegEx>

        <startOfTextStr>
            <h1 class="title">
        </startOfTextStr>

        <endOfTextStr>
            <P><div align="center">
        </endOfTextStr>
    </article>
</articleMetaData>

```

**Table 4.1.** RDF description

1. Article metadata: Metadata about the retrieved articles. This information includes the name of the source from where the article was located and the collection under which VIPAR classifies it.

2. Traversal directives: Specific actions for traversing a site. This includes the search depth limit (number of hops) from the root URL, and the number of minutes to wait between rechecking the site for new articles.
3. Traversal maps: The map starts with the root URL from which the agent is to begin a traversal of the site, and from which the agent can resolve relative URLs found at the site. A rule-based map of the pages of interest on the site is based on the URL structure of the site and is encoded via regular expressions.
4. Article delimiters: Markers to delimit the text of an article from other information on a given web page. (Banners and advertisements).
5. Article structuring rules: Rules for structuring the article text as XML. Regular expressions are used to reduce the various structural characteristics of an article, such as the title and author.

Based on this RDF ontology, a retrieval agent checks links found at an Internet newspaper site against the traversal map to determine if the article page is of interest. The agent checks with the VIPAR system to verify that the article has not already been incorporated. If the article is new, the agent retrieves the page, discerning the actual article text from the article delimiters, and cleans it of extraneous information on the page. The agent then marks up the clean text using XML, tagging the parts of the article (title, author, date, location, paragraphs, etc). The article is then posted to the VIPAR system. The agent continues to monitor the site based on the traversal directives, and posting new information, as it becomes available [26]. We have designed seven RDF

files that are currently used by VIPAR to collect and analyze bioinformatics data across the web.

### **4.3 Conclusions**

In the last couple of chapters we have transitioned from implementing a specific bioinformatics workflow case study to integrating other general domain specific solutions into SDMSWS architecture and expanding the bioinformatics environment. In particular we have demonstrated the feasibility of, SDMSWS as a framework for domain specific service based workflow composition and execution and at the same time, its role as a core element integrating diverse, domain specific specialized services that constitute the extended bioinformatics environment. In the following chapter we will perform a qualitative assessment of the SDMSWS framework, based on metrics derived for problem solving environments, against tools that were built on current web technologies and workflow principles, and in some cases serve the same purpose as SDMSWS. While some of these tools are commercial, others are products of research.

## 5 Framework Evaluation

### 5.1 Comparison Metrics

A problem-solving environment (PSE) is a computational system that provides a complete and convenient set of high-level tools for solving problems from a specific domain. The PSE allows users to define and modify problems, choose solution strategies, interact with and manage appropriate hardware and software resources, visualize and analyze results, and record and coordinate extended problem solving tasks. A user communicates with a PSE in the language of the problem, not in the language of a particular operating system, programming language, or network protocol [2]. A PSE should have the following properties:

- Support for all its user types, two principal ones being (1) application scientists interested primarily in using the PSE to solve a particular domain problem, (2) system developers who help achieve the objectives of the users [6].
- Support appliance-like use
  1. Provide the user with a comfortable and familiar interface.
  2. Abstraction must be used to hide the details of both the underlying computer system and the problem domain where appropriate [30].
  3. Problem-oriented to allow specialists to concentrate on their discipline, without having to become experts in computer science issues [31].

4. Analysis and presentation tool support. The use of graphics and visuals can enhance the usability of the PSE, for example, through tables and graphs to visualize the state of the application [56].
- Inter-operability and Integration
    1. Exchanging information across domains and among different providers is a very complex task. A PSE needs a thorough understanding of the diverse information sources and also certain expertise in them.
    2. To realize the full potential of network-supported workflows, we must enable the engineering of varied forms of integration suited to diverse architectures in practice.
    3. Use of standard technology defined and accepted by industry participants.
  - Collaboration
    1. Nowadays, many science and engineering projects are performed in collaborative mode with physically distributed participants. A PSE must include the ability to facilitate collaborative solutions.
    2. Human workflows and collaborations involve various elements of the participants changing situation and their activities.
  - Service Support
    1. Application development tools that enable an end user to construct new applications and support its execution on a set of resources.
  - A PSE must be dependent on, and support, the underlying network infrastructure.
  - It should be scalable. A PSE building tool must be able to add new functionality within its existent base.

- Adapt to changes, as the rate of change of experiment methodology is faster than the rate of change of tools available for performing analysis.
- Flexibility must exist in the ability to support a wide range of computational models that various domains may require.
- Support for debugging and exception handling.

These requirements hold true even for SDMSWS, which is categorized as a network-based framework providing problem-solving capabilities. To qualitatively evaluate the framework against both commercial and research based tools, we develop and use the PSE-based layering (Figure 5.1) of attributes namely, end-user issues, workflow issues, service issues, network-based issues, and other general issues as metrics.

End-User
Problem-Solving / Application Integration / Workflow
Underlying Technology and Service Implementation
Underlying Network Infrastructure

**Figure 5.1.** Problem Solving Environment Layering

The metric categories (and details) were chosen based on a survey of scientific workflow tools and user profiles relevant to a network-based PSE. The full list of the tools considered is given at the end of the thesis. Only a subset was used in the comparative

analysis. It is our firm belief that comparison metrics need to be sufficiently complete and comprehensive, and they need to have the scope that highlights both the PSE and the networking character of the solution, that is, it conforms to the software architecture layering shown in Figure 5.1. Thus, selected metrics describe important aspects of the requirements on the underlying network infrastructure, technology and services, application integration and workflows and the end user interfaces.

The author did not find sufficiently comprehensive open source tools, but did find a number of commercially available tools that were designed to help in some or all aspects of network-based problem solving. However, some open source workflow engines are briefly discussed in chapter 6. For this reason experimental, and quantitative, assessment of the tools was not possible for most part. Most tools examined had only published technical reports explaining their architecture and functionality. It was therefore decided to use more categorical and qualitative measurements in the comparative analysis. The metrics selected in each of the categories were based on the following reasoning:

- For End Users: The ideology of this framework had always been to reduce technological overhead on the end-users and allow them to concentrate on their research discipline, without having to become experts in computer science issues. This implied, in some ways, that we needed a solution that was easy to install on the user's system and principally required problem domain understanding for it to be used. It was also deemed necessary that the users have control of the interface and the services they needed to use.



- For Workflows: To provide flexibility, the developed system was designed to support manual or automatic selection of services from a list of published network based Web Services, and to use those selections to create and perform complex analyses.
- For Services: With scientific and business communities apparently moving towards the Web Services based approach, and with SDMSWS framework built on these principles, we thought it relevant to analyze and compare the underlying technologies, and their advantages and disadvantages.
- For Network Support: For universal accessibility, solutions and tools need to be available over the network supported by standard networking protocols. Also, to maintain the proprietary nature of the solutions it may be a requirement to keep the internal implementation details of their tools opaque to the end-user and secure from revealing its internals.
- For Other - General: Important issues while designing the workflow environment were to support platform independence, scalability by enabling integration of diverse and large data sets, analysis solution and methods to improve processing speed and optimize resource usage. Fault tolerance and exception handling was another concern as it is necessary to keep the user informed of failed processes and invocations and its also imperative to have provisions for detecting errors that were raised as a result of a user error or system failure.

The following section describes the measures used in each metric category and the qualitative scope of the measure.

### 5.1.1 Metric Definition and Scope

#### End-User Issues:

- **Abstraction:** Enable end-user to focus on their work instead of dealing with management and tuning of the tools that are not part of their workflow. This metric assesses the appliance-like nature of the tool. The value ranges from 1 to 5.  
1 – Complete abstraction  
5 – No abstraction
- **Usability:** Determined by the ease of installation on client machine and the required understanding to use the tool, also the extent of the functionality provided by the tool within the end-user domain. This was based on the subjective scale of 1 to 5 (rating provided by the author).  
1 – Easy to use with minimal understanding  
5 – Difficult to use and needs expertise
- **Human-Centric:** Support for human intervention during workflow execution.  
Full and Partial Support – Full (pro-active, e.g., resumable interrupts at will) or Partial (reactive, only pre-determined pause and intervention points) user control over workflow execution.  
No Support – No human intervention while workflow execution.

#### Workflow Issues:

- **Workflow Type:** States whether the workflow environment supports linear, non-linear workflows, or both.

- **Composition Tool:** The software or programming languages used to develop the workflow composition tool.
- **Workflow Automation:** Determines the support for automation of workflow execution and reuse of saved workflows.

#### Technologies / Services Issues:

- **Technology/Standards:** Lists the technologies involved in designing the framework and the protocols supported, if any.
- **Service Oriented:** States whether the tools are Web Service based or not.  
Yes – Web Service Based, No – Proprietary Services

#### Network-Based Issues:

- **Network-Based:** Current scientific computation tools are expensive, need expertise to use them, and hence have a restricted set of users. To provide universal access, applications need to be network accessible.  
Yes – Services can be invoked over the network.  
No – Does not support invocation of network-based services.
- **Standard Protocol Support:** Communication in SDMSWS is encapsulated in the Web Services protocols that run over standard network protocols (HTTP, TCP/IP) implying no change to the underlying network support.
- **Security:** The type of security provided to the applications and the framework. Frameworks can have a very basic web server based security or can have sophisticated enterprise level security.

General Issues (Open Source, Platform Support, Scalability, Processing and Fault Tolerance/Exception Handling):

- **Commercial:** Lists if the product is commercial or available in the public domain as freeware. The tool can also be available for research purposes.

Yes – Commercial

No – Available in public domain as freeware

- **Product Type:** Tells whether the product is a prototype or is a production-level release.
- **Open Source:** States whether the product is open source or not.
- **Platform Support:** Lists the platforms supported by the framework.
- **Scalability:** States whether other solutions and tools can be incorporated into the workflow environment.

Yes – Means that other public tools can be incorporated.

No – Indicates that it does not support inclusion of other solutions.

- **Processing:** States the use of any features to support higher processing. For example, use of state of the art processors or special design in the architecture to enhance the processing of tasks. It can either be limited or extensive based on the framework.
- **Fault Tolerance/Exception Handling:** States the extent to which exception handling is supported (limited or Extensive, through some special support), and if there are any explicit fault tolerant features or functionalities built into the system.

### 5.1.2 Survey of Workflow Tools

For the purpose of comparison we surveyed, among others, PSE's such as EDSS [53], Symphony [54], Coven [6] and GRASPARC [5]. These PSE's were designed to solve a special class of problems such as environment modeling and analyses, grids, visualization and high performance applications. Although some of the tools were network-oriented, there was no direct reference to the use of service based workflow composition and execution (as we know it today) in any of them. We also studied commercial workflow solutions from vendors in both science and business domains. Most of them were not available for us to evaluate. For example, we found that WebSphere Application Development Integration Edition [18] from IBM primarily focuses on support of Web Service based workflows in business-to-business industry. We also saw relevance in i-Flow [32], as it was a classical workflow environment with support for human centric drag and drop features for workflow composition. It has a Java-based architecture, which helped us develop some ideas on programming the workflow environment. These, along with some other graph based workflow notions were the motivation for developing a GUI based workflow editor. Also, right from the early stages of this research we have been concentrating on scientific workflows and tools for the life sciences discipline. We thus, narrowed our search to data and analysis solutions in the Life Sciences Domain. We performed extensive search on scientific workflows in various web resources and directories and found commercial and research driven solutions that were designed to support workflows in the life sciences discipline. However, only few of these had similar properties and goals as SDMSWS and thus were chosen for comparative analysis.

## 5.2 WebSphere Application Development Integration Edition

WebSphere Application Development Integration Edition (WSAD IE) [18] from IBM is a complete web application and services development environment. It provides the tools necessary for developing web applications that includes static web pages, JSPs, servlets and the XML deployment descriptor. It also is intended to be the ideal collaborative tool for the creation, assembly, publication, deployment and dynamic maintenance of Web Services using the latest Web Services standards such as SOAP, WSDL, UDDI and more recently BPEL4WS [58]. But the main focus of WSAD IE is on the interfaces coming from various clients and connectors used to connect to data and information resources in an n-tier enterprise software environment. It provides for wizards to generate interfaces described by WSDL. Services and service flows are an integral part of WSAD IE. A Service flow is a service composed of other services. In the WSAD IE terminology a smaller element of a workflow is called a micro flow, which represents a series of interactions required to do a single synchronous operation. Microflows are uninterruptible, short-lived operations that perform basic operations within a task rather than a complete task. There are different scenarios for which WSAD IE is used [18].

- A client accessing a service over SOAP. The tools in Application Developer Integration Edition can be used to create the service provider as well as the SOAP proxy used by service consumer to access the client.
- WSAD IE allows for services to be created from java beans. These services can also be available via the SOAP protocol.
- WSAD IE has tools for combining individual services into a single service flow.

### 5.2.1 Comparison Matrix

	SDMSWS	WSAD IE
<b>End-User Issues</b>		
Usability	1 (Easy to Install and Use)	5 (Need Expertise)
Human-Centric	Yes, Partial Support	Yes, Partial Support
<b>Workflow Issues</b>		
Workflow Type	Linear Scientific Workflows	Linear & Non-Linear Science and Business Workflows
Composition Tool	Swing Workflow Editor	Proprietary MicroFlow Constructor
Workflow Automation	Yes, Save and Reuse Workflows	Yes, Save and Reuse Workflows
<b>Service Issues</b>		
Technology/Standards	SOAP, UDDI, WSDL	SOAP, UDDI, WSDL
Service Oriented	Yes, Web Service Based	Yes, Web Service Based
<b>Network-Based Issues</b>		
Network Based	Yes	Yes
Security	Default Web Server Security for Services	Enterprise Level Security
<b>Other General Issues</b>		
Commercial	No	Yes
Product Type	Prototype, Research Based	Full Grown, Commercial
Open Source	Yes	No
Platform	All Platforms	Windows and Linux
Scalability	Yes, can incorporate any Web Service	Yes, can incorporate any Regular or Web Service

**Table 5.1.** SDMSWS Vs WSAD IE

Some of the issues that compare and contrast WSAD IE and the SDMSWS framework are:

- Support for Web Services: WSAD IE is a complete suit of tools (wizards) that helps create and deploy customized services that can run in its environment. In SDMSWS any network based service, described and deployed with Web Service protocols, can be incorporated as a component in the workflow.
- Support for Workflows: WSAD IE focuses on the shorter, synchronous, linear and non-linear workflows, which it calls microflows. SDMSWS has support for linear workflows, created from existing services. It also supports synchronous interactions between the tasks in a workflow.
- Save and Retrieve Workflows: In WSAD IE workflows can be saved and invoked at a later stage. SDMSWS workflow environment also allows for saving of workflows.
- Distributed Composition/Execution: In SDMSWS the workflow can have components that are distributed and can be invoked over the network. WSAD IE too has workflow components/tasks that are network based.
- Usability: WSAD IE needs certain amount of expertise and knowledge on the users part. SDMSWS needs lesser understanding on part of the user and is easy to adapt to.

WSAD IE is a comprehensive set of tools that provide support for tailored workflows in the business-to-business industry and the scientific domain. SDMSWS can be categorized as a subset of WSAD IE with some additional distinctive features.



## 5.3 i-Flow

i-Flow [32] is a workflow tool from Fujitsu Software that aids business groups to collaboratively plan, automate, track and improve business processes. Its a conventional software engineering workflow tool that defines the operational aspect of a business process, like sequencing of tasks and who performs them, the information flow to support the tasks, and the tracking and reporting mechanisms that measure and control them. It allows monitoring the flow and checking the execution of the workflow and keeps the history of workflow execution. Its primary usage is for definition and execution of human oriented workflows. Central to the architecture of i-Flow is a workflow engine. There are also adapters that can connect to databases, custom clients and other environments. The client architecture is delivered through a web browser. It is a java based workflow client tool that has features to construct workflows graphically.

### 5.3.1 Comparison Matrix

	SDMSWS	i-Flow
<b>End-User Issues</b>		
Usability	1 (Easy to install and use)	3 (Needs Expertise)
Human-Centric	Yes, Partial Support	Yes, Extensive Support
<b>Workflow Issues</b>		
Workflow Type	Linear Scientific Workflows	Linear & Non-Linear Business Workflows
Composition Tool	Swing Workflow Editor	Applet Workflow Editor

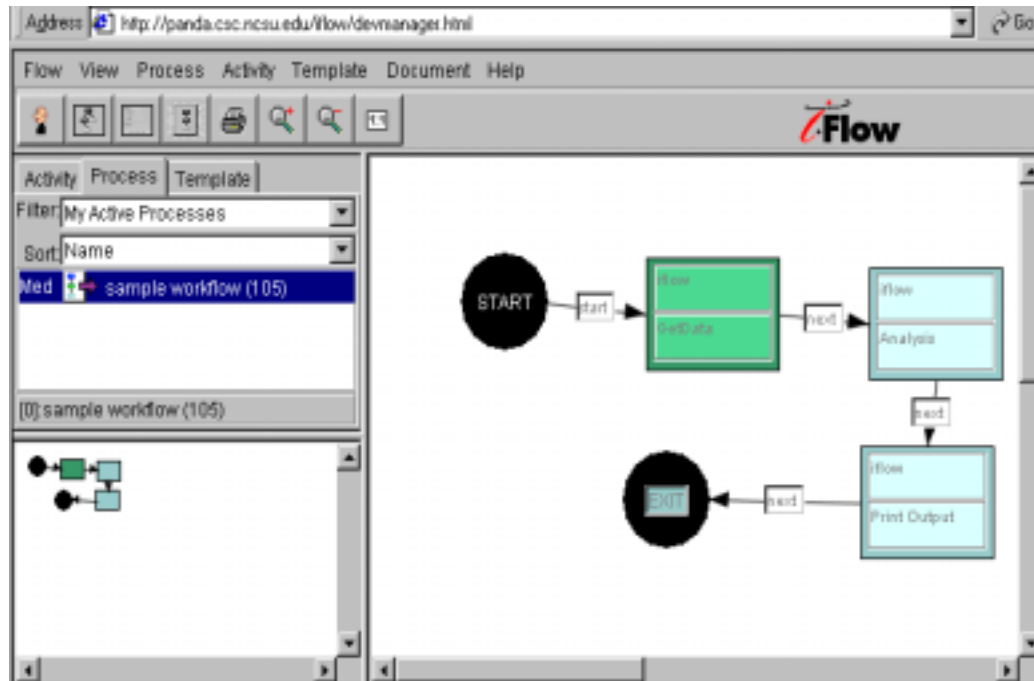
**Table 5.2.** SDMSWS Vs i-Flow

**Table 5.2.** (continued)

<b>Service Issues</b>		
Technology/Standards	SOAP, UDDI, WSDL	J2EE, RMI, IIOP
Service Oriented	Yes, Web Service Based	No support for services
<b>Network-Based Issues</b>		
Network Based	Yes	No, used mostly for Intranet based workflows
Security	Uses default Web Server	LDAP [41] Security
<b>Other General Issues</b>		
Commercial	No	Yes
Product Type	Prototype, Research Based	Full Grown
Open Source	Yes	No
Platform	All Platforms	Windows, Linux
Scalability	Yes, can incorporate any Web Service	Yes, Expansion through Use of Adapter Modules
Fault Tolerance	Limited Exception Handling	Extensive, Through use of handling modules
Processing	Handle limited Requests	High, Engine uses Numerous Processors

Following are some observations of i-Flow environment with respect to SDMSWS:

- Workflow Construction: i-Flow can be used to construct workflow templates (Figure 5.2), where components represent certain work to be performed. SDMSWS framework too has a workflow editor component with similar functionality that assists in workflow creation and execution.



**Figure 5.2.** i-Flow Workflow Environment

- Workflow Execution: i-Flow lacks the ability to invoke services on remote systems. SDMSWS workflow components can invoke remote services.
- Event Driven: In i-Flow on execution of a task a report can be generated and the concerned participant(s) can be informed of the same.
- Human-Centric: The workflow environment in i-Flow provides for human intervention and lets the user make changes during runtime. Similarly in SDMSWS the client can intervene the execution of the workflow at any stage.
- Saving Workflow Templates: Both i-Flow and SDMSWS allow the invocation of saved workflows, saving time for a periodically run workflow.

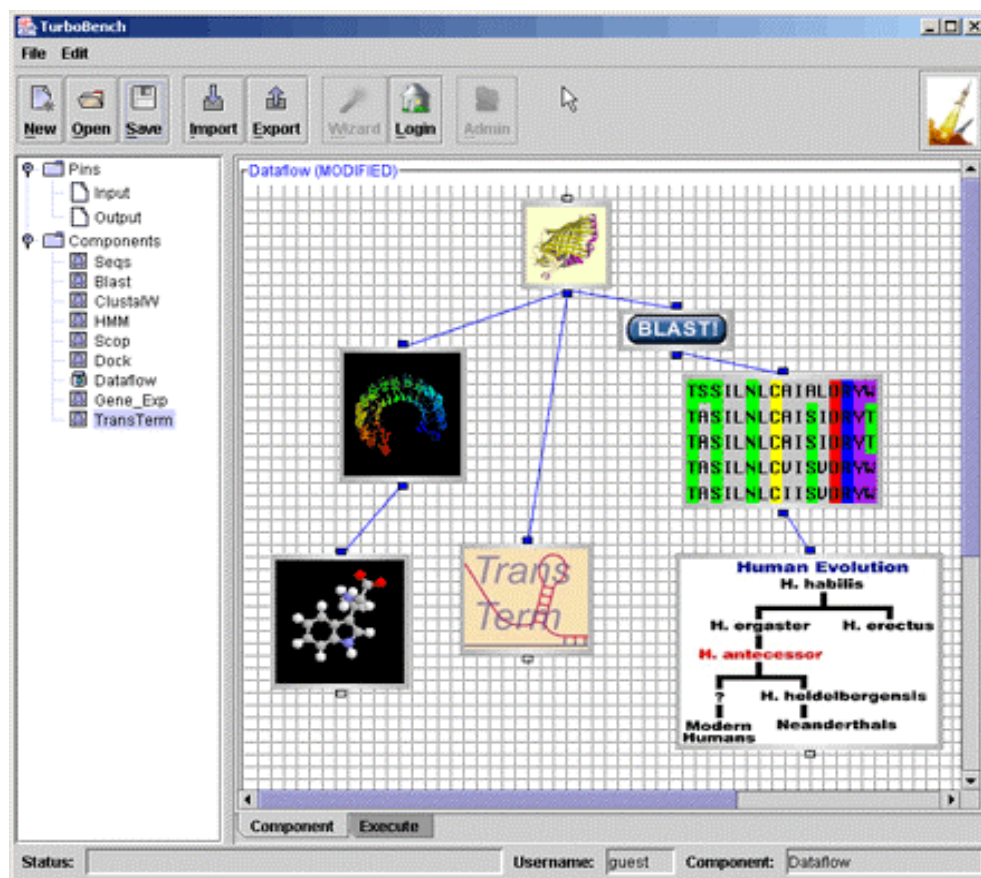
i-Flow is a tool supporting intra-organization workflows and lacks the ability to invoke remote services unless the architecture is customarily designed. Its comparison with SDMSWS was on the basis of the workflow related aspects they have in common.

## **5.4 TurboBench**

TurboBench [33] is a Java-based system that enables scientists to, (1) integrate standard bioinformatics applications and diverse data sources into computational workflows in which input data and results are passed from one step to the next and, (2) accelerate workflow execution through parallel computing to exploit heterogeneous collections of networked computers. Workflows correspond to graphs and are composed of components having well-specified sets of inputs and outputs and may perform computations or other operations on data. There are atomic components (self-contained operations such as running a Java or a Perl program and SOAP services in its latest release) and workflow components (links components together, with each component executing one after the other in a sequence). Components are structured XML objects containing information required to manage and execute them. Users can create atomic components using wizards that produce XML objects automatically by generating appropriate wrappers [33]. Its architecture consists of:

- **Master:** The Master is responsible for overall management of the system. It authenticates users, coordinates and monitors the execution of individual jobs, handling errors, and notifying Clients when the job is complete.

- **Client:** Clients are used to create atomic components or workflows, store them and submit them for execution to the system. Clients are run on desktop machines and include wizards and the TurboChart tool (Figure 5.3).
- **Workers:** The Workers are the machines that perform the real computational work. They execute components, access databases, and transform data as it passes from one component to the next. Each Worker runs a small program (called a daemon) that decides exactly what the Worker should do. It takes care of data manipulations, conversions, and communications required for setting up the task. It executes the actual component to carry out the required computation or data access. Finally, it returns information to the Master when the task completes.



**Figure 5.3.** TurboBench Workflow Environment

### 5.4.1 Comparison Matrix

	SDMSWS	TurboBench
<b>End-User Issues</b>		
Usability	1 (Easy to install and use)	5 (Needs Expertise)
<b>Workflow Issues</b>		
Workflow Type	Linear Scientific Workflows	Linear & Non-Linear Scientific Workflows
Composition Tool	Swing Workflow Editor	Swing Workflow Editor
Workflow Automation	Yes, Save and Reuse WF	Yes, Save and Reuse WF
<b>Service Issues</b>		
Technology/Standards	SOAP, UDDI, WSDL	EJB'S, XML, Databases Yes, Proprietary (XML) & SOAP Service Support
Service Oriented	Yes, Web Service Based	
<b>Network-Based Issues</b>		
Network Based	Yes	Yes
Security	Uses default Web Server	Enterprise Level Security
<b>Other General Issues</b>		
Commercial	No, Research Use	Yes
Product Type	Prototype	Full Grown
Open Source	Yes	No
Platform	All Platforms	UNIX, Linux, Windows
Scalability	Yes, can incorporate any Web Service	Yes, Plug-and-Play Arch.
Fault Tolerance	Moderate Exception Handling	Guaranteed / Automatic Rerun of Tasks
Processing	Limited Requests	High, Scheduling Models For high performance

**Table 5.3.** SDMSWS Vs TurboBench

Following are some observations of TurboBench with respect to SDMSWS:

- **Support for Workflows:** TurboBench combines atomic components to form linear and non-linear workflows. Atomic components correspond to self-containing operations such as executing a java method. Workflow components in SDMSWS represent Web Services.
- **Service Components:** TurboBench supports only those applications in workflows that are customized to run in its environment unlike SDMSWS workflow editor, which can incorporate any network-based service that supports standard Web Service descriptions and protocols.
- **Network Based Workflow Execution:** TurboBench workflow can consist of components running on any machine in a network. In SDMSWS also workflows can consist of tasks distributed over the network.
- **Parallel Computing:** An application execution engine provides run time support for parallel computations in TurboBench. SDMSWS workflow engine currently does not support parallel computation of tasks.
- **Save/Retrieve Data:** TurboBench has a data repository service that aids data to be stored in different formats. For example data can be stored in files or databases. In SDMSWS data can be stored on client side as files.
- **Processing:** Use of scheduling model in TurboBench facilitates for load balancing among available resources. In SDMSWS the workflow component is very primitive and does not support such capability.

Overall, the TurboBench workflow environment provides extensive support for life sciences based workflows, but requires workflow components to be tailor-made to work in its execution environment.

## **5.5 Visual Integrated Bioinformatics Environment (VIBE)**

VIBE [34] is a visual programming interface to VIBE servers, which provide access to many common bioinformatics algorithms. The graphical interface provides an extensible drag-and-drop environment for the creation of sequence analysis pipelines and visualization of results. The VIBE interface comes with a set of analyses that can be pipelined within the interface [34]. VIBE was developed by INCOGEN, Inc. in collaboration with TimeLogic Corporation. The VIBE architecture is based on J2EE object oriented architecture standards and RMI technology to provide a scalable and extensible system. VIBE provides the following features [34]:

- Efficient drag-and-drop sequence analysis pipeline construction from a wide selection of tools and algorithms.
- Users can either run the tools with their default parameters or change them.
- Design time checking of workflow specification.
- Pipelines can be saved as XML specifications and can be invoked at a later stage. User can also edit the XML specifications.
- Support for specialized visualization tools for certain analyses. Data can also be viewed as XML or plain text.



### 5.5.1 Comparison Matrix

	SDMSWS	VIBE
<b>End-User Issues</b>		
Abstraction	2 (Yes, Abstracts Details)	2 (Yes, Abstracts Details)
Human-Centric	Yes, Partial Support	Yes, Partial Support
<b>Workflow Issues</b>		
Workflow Type	Linear Scientific Workflows	Linear & Non-Linear Scientific Workflows
Composition Tool	Swing Workflow Editor	Proprietary Editor
Workflow Automation	Yes, Save and Reuse WF	Yes, Save and Reuse WF
<b>Service Issues</b>		
Technology/Standards	SOAP, UDDI, WSDL	Multi-tiered J2EE and RMI
Service Oriented	Yes, Web Service based	No, Proprietary Services
<b>Network-Based Issues</b>		
Network Based	Yes	Yes
<b>Other General Issues</b>		
Commercial	No, Research Use	Yes
Product Type	Prototype	Full Grown
Open Source	Yes	No
Platform	All Platforms	All Platforms
Scalability	Yes, can incorporate any Web Service	Yes, use of Multi-Tiered Architecture
Processing	Limited Requests	High, Use of Application Layer Distribution

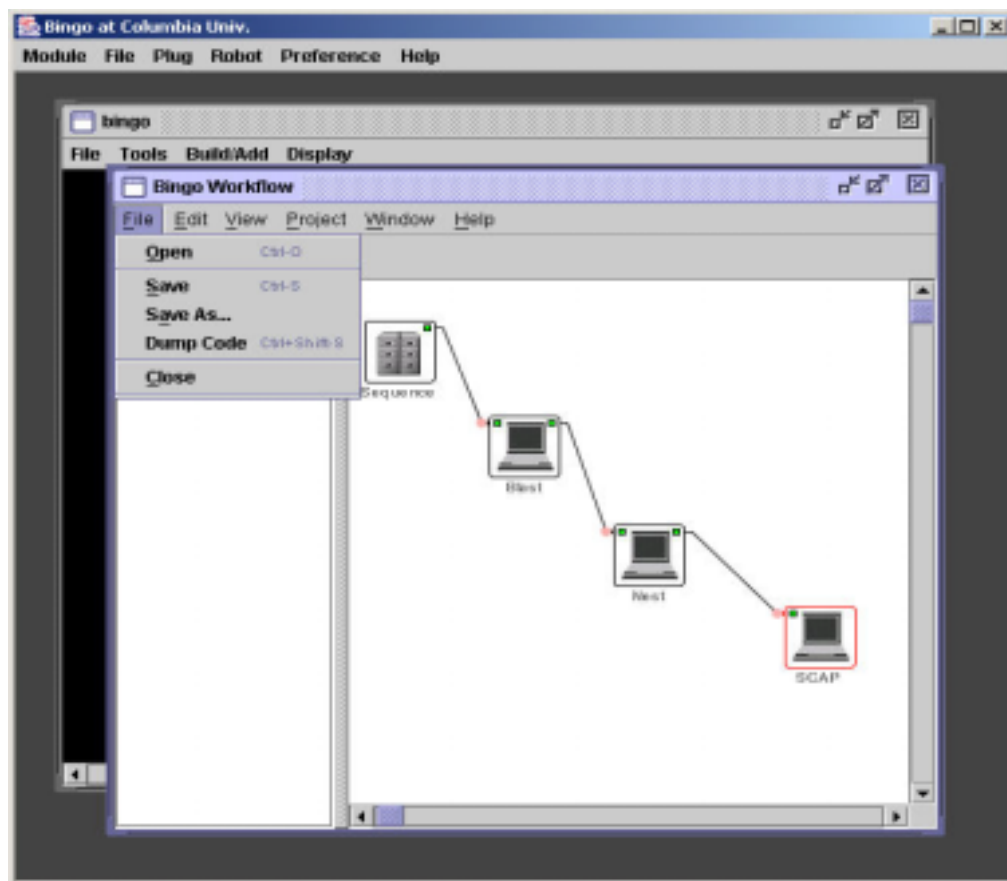
**Table 5.4.** SDMSWS Vs VIBE

Some comparison issues regarding VIBE and SDMSWS are:

- **Workflow Verification:** VIBE provides design time checking to assist users create valid workflows and reduce the probability of run-time errors [34]. SDMSWS workflow currently does not provide for any workflow verification.
- **Retrieving Saved Workflows:** In VIBE workflows can be saved as XML with the client. It can be invoked later and used with different inputs. SDMSWS workflow editor can save workflows in plain text format, which can be invoked at a later time.
- **Service Components:** VIBE supports only those applications in workflows that are customized to run in its environment. SDMSWS workflow editor can incorporate any network component based component as long as the component supports standard Web Service protocols and descriptions.
- **Visualization:** VIBE has special applications and libraries, which provide better visualization of data. In SDMSWS data format is dependent on the type supported by the service.
- **Scalability:** VIBE is based on J2EE, RMI technology and has a multi-tiered architecture which allows it to be scalable and extensible. SDMSWS workflow editor currently has a primitive architecture that provides support for limited number of services.
- **Access to Data:** Results of workflow runs are stored in the database and can be retrieved later. In SDMSWS results are stored as files on the client side.
- **Processing:** VIBE has the ability to distribute the application layer in the architecture to improve system performance [34].

## 5.6 BINGO

BINGO [35] is a three-dimensional graphical environment for sequence analysis and structure modeling. The purpose of BINGO is to develop a platform where software tools can communicate with each other by streamlining their inputs and outputs. It offers a platform for integrating bioinformatics software tools and databases with graphic analysis. The software tools are connected with various databases to automate complex computations, where the results can interact in real time with graphical analysis system. Different workflow threads can be created in BINGO workflow environment.



**Figure 5.4.** BINGO Workflow Environment

The architecture consists of:

- BINGO workflow: Used to create complex job threads. Each node in a chart is a scientific software tool. The input and output of different nodes can be connected so that the output from one program can be used as input for another. The job thread can be saved and invoked later.
- JGRASP: JGRASP [35] is for three-dimensional visualization of molecular structure and sequences. With the built-in Java version of Delphi, JGRASP runs on any machine including applets from web pages.
- Alignment Editor: This component is for visualization of multiple sequence alignment in several sequences formats.

### 5.6.1 Comparison Matrix

	SDMSWS	BINGO
<b>End-User Issues</b>		
Usability	1 (Easy to install and use)	4 (Needs Expertise)
<b>Workflow Issues</b>		
Workflow Type	Linear Scientific Workflows	Linear Scientific Workflows
Composition Tool	Swing Workflow Editor	Java-Based Editor
Workflow Automation	Yes, Save and Reuse WF	Yes, Save and Reuse WF
<b>Service Issues</b>		
Technology/Standards	SOAP, UDDI, WSDL	Java, Databases
Service Oriented	Yes, Web Service based	No, Proprietary Services

**Table 5.5.** SDMSWS Vs BINGO

**Table 5.5.** (Continued)

<b>Network-Based Issues</b>		
Network Based	Yes	No
<b>Other General Issues</b>		
Commercial	No, Research Use	No, Research Use
Product Type	Prototype	Prototype
Open Source	Yes	Yes
Platform	All Platforms	SGI 6.5, Intel Linux and Sun Solaris

Some comparative issues between SDMSWS and BINGO are:

- Service Components: BINGO supports only those applications in workflows that are customized to run in its environment. It uses the tools available in the Jackal Package [35]. SDMSWS workflow editor can incorporate any service based on standard Web Service protocols and descriptions.
- Visualization: BINGO has visualization modules that can be used for three-dimensional view of data.
- Workflow Save and Retrieve: In BINGO workflows can be saved and invoked later saving time for repetitive invocation of the same workflow. SDMSWS workflow editor has similar features.
- Scalability: Scalability in BINGO is supported by providing modules to plug-in new components in the workflow environment.
- Platform Support: The strength of BINGO is its visualization feature, which is supported well by the processing capability of SGI machines.

## **5.7 Workflow based Architecture to support Scientific Applications (WASA)**

The WASA [36] project was aimed at providing a workflow based integrated environment for advanced applications in a variety of domains, including life sciences. The first WASA architecture (1994-1996) was based on layers of distinct functionality, each of which consisted of a set of components. The defined layers were, The User Interface layer (for designing workflows, data manipulation, visualization and runtime monitoring), Internal Tool Layer (for workflow engine), Enhanced Database Functionality Layer (for advanced functionality in system-independent way) and the Database Layer. The second WASA prototype (1994-1996) was based on client-server architecture. The workflow engine was a multi-threaded Java program that read workflow specifications from a relational database, instantiated workflows and controlled the execution of workflows. Users accessed the workflow management system using standard Web Browsers as workflow clients. Also, workflows contained sub workflows. When a complex workflow was launched, only that sub workflow was started that had to go first, subsequent ones were only retrieved from the database when needed. The latest prototype architecture is based on CORBA (Common Request Broker Architecture) [37] and java with features like workflow modeling and execution, reuse of workflow schemas and distributed workflow execution. Workflows are implemented as CORBA objects, which communicate with each other by sending and receiving messages using an object request broker. The use of object-oriented technology achieves interoperability between applications that are components of a workflow system.

### 5.7.1 Comparison Matrix

	SDMSWS	WASA
<b>End-User Issues</b>		
Usability	1 (Easy to install and use)	1 (Easy to install and use)
Human-Centric	Yes, Partial Support	Yes
<b>Workflow Issues</b>		
Workflow Type	Linear Scientific Workflows	Scientific Workflows
Composition Tool	Swing Workflow Editor	Swing or Applet Editor
Workflow Automation	Yes, Save and Reuse WF	Yes, Save and Reuse WF
<b>Service Issues</b>		
Technology/Standards	SOAP, UDDI, WSDL	CORBA, Databases
Service Oriented	Yes	No, CORBA Objects
<b>Network-Based Issues</b>		
Network Based	Yes	Yes
<b>Other General Issues</b>		
Commercial	No, Research Use	Yes
Product Type	Prototype	Prototype
Open Source	Yes	No
Platform	All Platforms	All Platforms
Scalability	Yes, can incorporate any Web Service	Yes, use of CORBA Middleware architecture to support scalability

**Table 5.6.** SDMSWS Vs WASA

Following are some comparative issues between WASA and SDMSWS:

- **Workflow Execution:** In WASA the CORBA infrastructure allows the assignment of workflow objects to different servers. Parts of workflow can communicate and exchange messages. This allows controlling workflows in a fully distributed way without the need of a centralized workflow engine. SDMSWS has a central workflow engine that governs the execution of a workflow.
- **Workflow Components:** WASA supports only those applications in workflows that are specific technology (CORBA) based.
- **Workflow Clients:** Web browsers interpreting java applets or stand-alone java applications are used as workflow clients. SDMSWS uses the swing workflow editor as the workflow client.
- **Application Integration:** In WASA support for integration of domain specific applications is based on a common interface definition (Interface definition Language) standard. In SDMSWS the use of standard interoperability protocols and technologies leverage the task of domain specific application integration.
- **Runtime Modification:** Using explicit modeling of states of workflow instances, WASA can implement dynamic runtime modification of workflow parameters [36]. SDMSWS workflow environment allows for modification of workflow component attributes during the run time of the workflow at break points, which occur in between the tasks.
- **Architecture Scalability:** Scalability in WASA is supported by the use of CORBA middleware architecture, which accommodates various CORBA objects.



## 5.8 GeneFlow

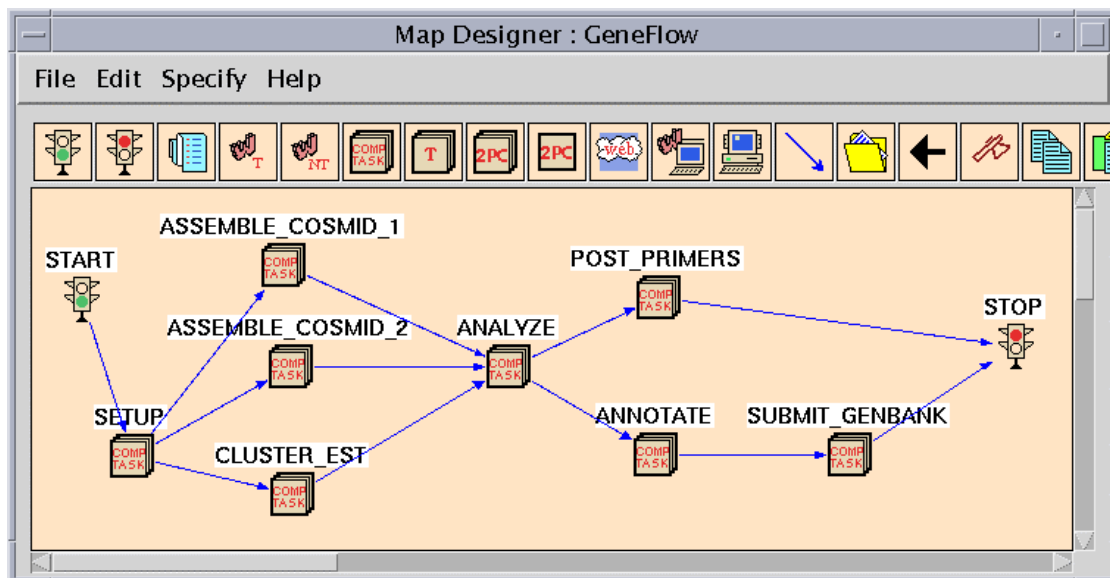
GeneFlow [38] is a scientific workflow tool based on the METEOR (Managing End-To-End Operations) [39] workflow management framework. It is a workflow system for automating many aspects of genome data analysis in a collaborative research environment. The system is designed to leverage the power of the Web to tie together the various activities of data generation and analysis at multiple facilities. GeneFlow consists of a number of software modules that carry out a particular function or set of functions. These modules are being developed in Java and Perl programming languages. The modules are designed such that it can be integrated into workflows using the METEOR workflow management system.

Workflow management techniques developed in the METEOR project at the LSDIS lab are intended to reliably support large-scale, complex and adaptive workflow applications in real-world multi-enterprise heterogeneous computing environments [38]. It coordinates, controls, monitors workflows consisting of automated/semi-automated tasks and the information flowing between them. It addresses several open issues like transactional workflows, interoperability via CORBA and/or Java, familiar easy-to-use interfaces (Web browsers), easy-to-use graphical design tool, and multi-paradigm workflows. Few of the technical capabilities of METEOR include [39]:

- Comprehensive modeling supporting by METEOR's workflow model with its approach to specify human and automated tasks and intertask dependencies, and a comprehensive graphical designer.

- Scalability and robustness, enabled by fully distributed architecture and scheduling.
- Rapid development of complex applications for dynamic businesses, enabled by automatic code generation from the specification provided graphically. Support for enterprise wide and multi-enterprise environments.
- Basic interoperability, security and exception handling and recovery

METEOR can be used to create workflows visually, by using a graphical "drag and drop" utility. METEOR is web enabled giving access to the components of the workflow. Figure 5.4 depicts the METEOR graphical workflow building utility showing a number of GeneFlow modules linked together in a workflow. Each component is a module performing a task.



**Figure 5.5.** GeneFlow Workflow Environment

### 5.8.1 Comparison Matrix

	SDMSWS	GeneFlow
<b>End-User Issues</b>		
Human-Centric	Yes, Partial Support	Yes, Partial Support
<b>Workflow Issues</b>		
Workflow Type	Linear Scientific Workflows	Linear & Non-Linear Scientific Workflows
Composition Tool	Swing Workflow Editor	METEOR Workflow Wfms
Workflow Automation	Yes, Save and Reuse WF	Yes, Save and Reuse WF
<b>Service Issues</b>		
Technology/Standards	SOAP, UDDI, WSDL	Java, CGI, Perl
Service Oriented	Yes, Web Service based	No
<b>Network-Based Issues</b>		
Network Based	Yes	Yes
<b>Other General Issues</b>		
Commercial	No, Research use	Yes
Product Type	Prototype	Prototype
Open Source	Yes	No
Platform	All Platforms	All Platforms

**Table 5.7.** SDMSWS Vs GeneFlow

Following are some comparative issues between SDMSWS and GeneFlow:

- Workflow Construction: GeneFlow can be used to construct workflow templates (Figure 5.2), using the METEOR Wfms, where components represent certain

work to be performed. SDMSWS framework has a Swing based workflow editor component with similar functionality that assists in workflow creation and execution.

- **Service Support:** Modules supported in the workflow are java and Perl based and are designed to work in the METEOR workflow environment.
- **Non-Linear Workflows:** Geneflow supports construction of both linear and non-linear workflows. The current SDMSWS workflow tool supports only linear workflows.
- **Network Based Workflow Execution:** GeneFlow workflow can consist of components running on any machine in a network.

## **5.9 Bioinformatics Workflow and Data Management System (GeneBeans)**

GeneBeans [40] is a problem-solving environment for mining databases. The goal of the project is to, (1) improve the usability and increase the functionality of bioinformatics query systems and (2) provide tools by implementing a graphical dataflow-programming interface. The system lets biologists deal with high-level components that can be connected to form queries. There is support for branching networks of nodes, as well as traditional pipeline constructions. The GeneBeans system uses a three-layer architecture: (1) user interface, for workflow composition of components that perform specific querying on the database, (2) a dataflow engine that executes commands on data retrieved from the database and (3) a database (or several federated data sources). Session beans combined with database support decide exactly what data and algorithms are used to run

a particular experiment. They will efficiently run a new analysis that is a variant on one previously computed, without recomputing intermediate results. EJBs provide for resource sharing, increasing the performance of a tool that makes many short-lived connections to a database. They support a modular architecture, which will extend the servers with new components to be composed into analyses [40].

### 5.9.1 Comparison Matrix

	SDMSWS	GeneBeans
<b>End-User Issues</b>		
Human-Centric	Yes, Partial Support	Yes, Partial Support
<b>Workflow Issues</b>		
Workflow Type	Linear Scientific Workflows	Linear & Non-Linear Scientific Workflows
Composition Tool	Swing Workflow Editor	Applet Workflow Editor
Workflow Automation	Yes, Save and Reuse of Workflows	Yes, Save and Reuse of Workflows
<b>Service Issues</b>		
Technology/Standards	SOAP, UDDI, WSDL	EJB's, Database WebSphere App Server
Service Oriented	Yes, Web Service based	No, Proprietary Tools
<b>Network-Based Issues</b>		
Network Based	Yes	No, Intranet-Based

**Table 5.8.** SDMSWS Vs GeneBeans

**Table 5.8.** (continued)

<b>Other General Issues</b>		
Commercial	No, Research Use	No, Research Use
Product Type	Prototype	Prototype
Open Source	Yes	No
Platform	All Platforms	All Platforms

Following are some comparative issues between SDMSWS and GeneBeans:

- Usability: Provides a graphical dataflow interface that shields technological details from end users. SDMSWS is based for similar purposes.
- Non-Linear Workflows: GeneBeans supports construction of non-linear workflows.
- Workflow Components: In GeneBeans nodes in a workflow are connected to form complex database queries. The query modules are designed and developed to execute only within the GeneBeans workflow editor.

## 5.10 Conclusions

Most of the tools and products discussed in this chapter either support scientific workflows or assist document exchange and application integration through workflows in the business-to-business industry. Their comparison with respect to SDMSWS was based on the functionality they have in common.

The most significant observation, among others, resulting from the comparative analysis reveals that our framework minimizes the need for customization and integration of new component workflow services, and it shows that, unlike commercial solutions, it is relatively independent of the service domains and can incorporate any network-based service that conforms to standard Web Services description and protocols. WSAD IE supports Web Service based workflow composition but the participating components are serviced within its framework. All other tools have proprietary services that are dependent on the specific environment they were composed to run in. Other advantages of the SDMSWS workflow environment are that it is platform independent, open source and available in the public domain as freeware.

It is the finding of this work that complex distributed scientific workflows can be, and should be, supported using open-source service-based solutions. However, based on these factors it would be premature to conclude on the effectiveness and contribution of SDMSWS framework to the scientific community. It has many open research issues, as discussed in the following chapter, that when investigated and incorporated with the current infrastructure, we believe, will contribute substantially to the use of Web Services by the scientific research community.

## **6 Conclusions**

This thesis studied the feasibility, usability and effectiveness of the Scientific Data Management Service Workflow System (SDMSWS) architecture, which eases the access to scientific data and solutions, and manipulation and analysis of the same, with an interactive GUI-based workflow construction and execution environment that supports the Web Services framework. The work illustrates how the defined architecture extends by facilitating integration of diverse domain specific components. Finally, the framework assessment was performed based on problem solving environment metrics, and SDMSWS was compared against commercial standards and research solutions driven by similar goals. We first present a summary of achievements followed by an overview of the open research issues. We end this chapter with a brief overview of some technologies that are representative of the current state of Web Services and bioinformatics research.

### **6.1 Summary**

We developed the SDMSWS framework to achieve two goals: (1) to have a workflow construction and execution framework that creates workflows using the services registered with the registry and (2) to use the framework as a means of extending the bioinformatics environment, integrating diverse domain specific specialized services. To achieve these goals, research tools developed by scientists are registered as services with the UDDI registry. The idea of having a service registry that holds the details about services is to reduce the overhead on the scientists of searching for data acquisition and



analysis resources. Service descriptions such as methods implemented, input and output parameters and other details are described in a WSDL file that is published to the UDDI registry. Any organization interested in using a service can bind to it using the information available at the UDDI and the WSDL description of the service. This also allows for easy and timely update of the service descriptions that can propagate to all users quite rapidly. The GUI-based workflow construction and execution editor lets the end user define tasks and associate them with services. The user inputs the necessary information regarding the service. Once the workflow starts executing, each task (service) reads the parameters, creates a SOAP call and invokes the appropriate service. Workflows could be of the two types, first where human intervention is needed to analyze the input and output parameters, for some tasks to execute, and second where the workflow executes the tasks in the sequence from start to stop without any interruption. We were successfully able to implement a set of services constructed for a bioinformatics workflow. We also tested, in a limited way, the extensibility of the framework by describing and integrating other domain specific tools.

## **6.2 Future Developments**

There are a number of additional features and architectural developments that could extend and evolve SDMSWS framework:

- The most pressing task is to develop a domain independent, robust workflow engine that will incorporate workflows from all areas of scientific research. In

particular there is immediate need for supporting composition and execution of complex workflows.

- Incorporate a standard formal language or methodology to support description, sequencing and ordering of workflow components.
- The current workflow framework does not support validation and verification of the workflow description. This can be incorporated using workflow grammars like Web Services Flow Language (WSFL) or XLANG, or a combination of both.
- Provide documentation of the workflow for better understanding of the domain science issues.
- Possible description of a generic abstract workflow specification language that can be mapped and converted to an executable format. [Work in progress at San Diego Super Computer Center as part of the SciDAC [52] project].
- Import other Genetics and Bioinformatics workflow examples into the SDMSWS system, adapt to fit, and generalize.
- Build and maintain a library of complex workflows that are designed over time.
- Incorporate the construction and execution of complex works not just linear. Workflows that are like acyclic graphs.
- Optimize the resource utilization and perform load balancing by parallelizing the execution of tasks/services on numerous processors.
- Provide storage, query, retrieval and security of data, both archive and current, that is generated as part of the workflow execution.

In the implementation and experimentation domain the work that needs to be done is to field test SDMSWS architecture and implement workflows from diverse areas of scientific research. Such experiments in other areas will help determine the true applicability, scalability and usefulness of the SDMSWS framework.

## **6.3 Latest in Technology and Possible Research Areas**

### **6.3.1 Open Grid Service Architecture**

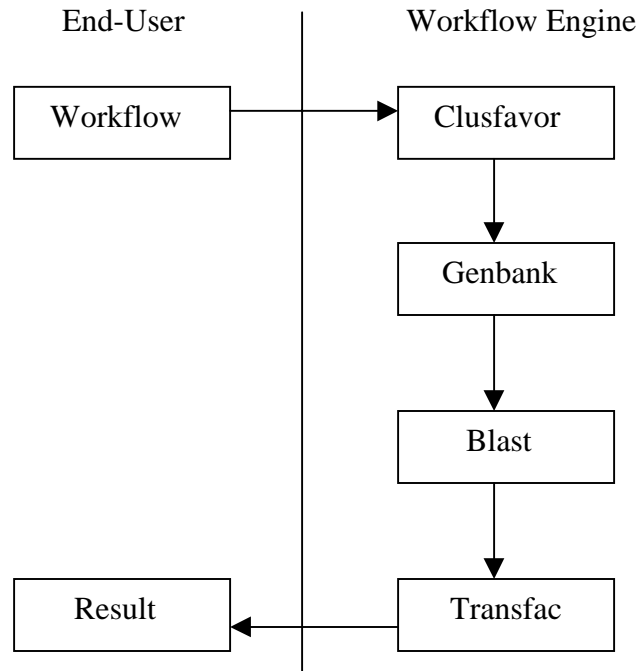
One of the architectures that uses the Web Services technology is the Open Grid Service Architecture (OGSA) [43] architecture. Open Grid Service Architecture integrates services across distributed and heterogeneous organizations. This integration is challenging because of the need to achieve various qualities of service (QOS) when running from various native platforms. The OGSA is based on the concepts and technologies from the GRID [42] and Web Services. This architecture defines uniform exposed service semantics (the grid service) and defines standard mechanisms for creating, naming and discovering transient Grid services instances. It provides location transparency and multiple protocol bindings for service instances and supports integration with underlying native platform facilities [43]. In OGSA, Grid technology is aligned with Web Services technologies to achieve service description and discovery and automated generation of client and server service descriptions. It also defines, in terms of WSDL, interfaces and mechanisms required for creating and composing distributed systems.

Service bindings can support reliable invocation, authentication, authorization, and delegation.

### 6.3.2 Web Services Flow Language (WSFL)

The goal of Web Services workflow is to enable integration across distributed environments and organizations that make use of many Web Services. The Web Services Flow Language (WSFL) is a proposed standard from IBM that addresses workflows on two levels [44]:

- It supports a directed-graph model approach to defining and executing processes.
- It defines a public interface that allows processes to advertise themselves as Web Services.



**Figure 6.1.** Flow Model

Figure 6.1 provides a basic example of a simple scientific workflow process discussed in chapter 3. It is an example of a directed-edge graph where lines connecting activities indicate the flow of processing control between activities. The workflow engine can incorporate WSFL and process tasks in the workflow in a defined sequence. This will help defining the sequence and ordering of tasks upfront. WSFL could be a potential substitute for the current native workflow engine language in the SDMSWS workflow editor. In a recent development in the area of workflow grammars, IBM and Microsoft have collaborated to propose a workflow language called BPEL4WS [58], which is conceptualized in ways from WSFL and XLANG [47].

### **6.3.3 Web Services Conversation Language (WSCL)**

As part of the continuing work based on the extension of Web Services technologies, comes another XML based workflow definition grammar called Web Services Conversation Language. “WSCL allows the abstract interfaces of Web Services, implying business level conversations or public processes supported by a Web Service, to be defined. WSCL specifies the XML documents being exchanged, and the allowed sequencing of these document exchanges. WSCL conversation definitions are themselves XML documents and can be interpreted by Web Services infrastructures and development tools. It can work in conjunction with other service description languages like WSDL. It could also be used to provide protocol-binding information for abstract interfaces, or to specify the abstract interfaces supported by a concrete service. The

purpose of WSCL is to provide and define the minimal set of concepts necessary to specify conversations” [45].

WSCL supports a formal approach to describe business level conversations, which is based on XML, with the expectation that it will be extended for more complex Web Services frameworks involving multiple participants. A key advantage of WSCL over other formal specification languages is that it facilitates the creation of service frameworks that will enable service implementers to offload the responsibility for conversation-based tasks to the underlying implementation. A service developer could create WSCL specifications documenting their service’s abstract interface and make it available to potential users. A software developer could then use WSCL-compliant Web Services servers and tools that provide document type validation, conversation tracking, and message dispatching to the application logic, and other such functionality. Web Services developers will be supported by tools that will allow them to map the interactions outlined in the conversation definition to any existing application [45].

#### **6.3.4 BizTalk**

BizTalk [46] a product from Microsoft primarily focuses on enabling businesses to exchange data from disparate data sources, standards and technologies. The exchange of data is dependent on a framework that is based on industry standard XML. Some features that BizTalk provides to the business community are [46]:

- **Enterprise Application Integration:** It enables interoperability between applications within an enterprise making it possible for companies to have existing applications communicate with each other without the need to completely rewrite those applications.
- **Business-to-Business Integration:** It provides for businesses to communicate and send documents to each other in a variety of means, such as Hypertext Transition Protocol (HTTP), Secure Sockets Layer (SSL) and File Transport Protocol (FTP).
- **Process Orchestration:** Business analysts can define the business process of how the communication must take place All of the modeling of the business process is defined and compiled into a special XML document, known as the XLANG [47].
- **Workflow Development Environment:** Workflow represents a process involving a number of participants, within an organization, exchanging information.

### **6.3.5 Open Source Workflow Projects**

- **OSWorkflow:** OSWorkflow [59] is an open-source workflow system that allows users to create and deploy workflows. It can be considered a low-level workflow implementation. For example, loops, conditions, and processes or tasks that might be represented by a graphical icon in other workflow systems must be coded in OSWorkflow using some scripting language. An XML description that conforms to the OpenSymphony [59] Document Type Definition (DTD) is translated into a workflow process. The workflow engine uses this description to execute processes or tasks.

- **OFBiz:** OFBiz (Open for Business) [60] is a comprehensive open source project for doing business online. This product has tools to compose and execute business workflows. OFBiz uses XML Process Description Language (XPDL) [62] as its process definition language. XPDL is an open standard.
- **OBE (Open Business Engine):** The Open Business Engine [61] is an open source workflow engine written in Java. It provides an environment for executing activities in a controlled and centralized environment. It consists of many components namely, OBE core for support libraries, OBE runtime engine that executes workflows, OBE client API, OBE designer for composing workflows and OBE servers that control the complete workflow environment.

Certain functionalities of these open source workflow projects look promising, and can be incorporated in parts in the development of workflow environments like the SDMSWS.

### **6.3.6 Bio\* projects**

Bio\* projects [49] is a series of freely available open-source projects, in which software engineers have developed re-usable code libraries in different programming languages like Perl, Java, Python and LISP. These libraries automate common bioinformatics tasks and provide methods for importing and exporting data between data sources. For example, to fetch a piece of data from a database, the bioinformaticists use the Bio\* libraries to do the fetch, put the information in a standard format, and return the



reformatted data to their script. The Bio\* libraries though have not solved the problem of the brittleness of online data sources. As soon as the structure of the web pages changes, the Bio\* library fails and has to be patched up using the adapter modules available for each of the online resource [50].

### **6.3.7 Discovery Link - IBM's Life Sciences Framework**

IBM's, Discovery Link [51] solutions is based on open standards including Web Services standards like Simple Object Access Protocol (SOAP), XML, WSDL, and databases, which allow services and products from multiple vendors to work together to form comprehensive life sciences computing solutions. It promotes efficient application development through component technology and code reuse decreasing the amount of time and resources necessary to develop new applications. Applications can run on open, standards-based systems. This framework provides for knowledge management, data integration, high-performance computing and storage services needed for life sciences research laboratories. It provides [51]:

- Knowledge management tools to transform data into information.
- Data integration tools for analyzing information extracted across varied resources and diverse data domains.
- High-performance computing for modeling, simulation and visualization.
- Storage and retrieval technologies.
- Support and improved performance for scientific workflows.

## References

- [1] Enrique Castro “A perspective on Web Services”, WebServices.Org, 18/02/2002.  
<http://www.webservices.org/index.php/article/articleview/113/1/61>
- [2] E. Gallopoulos, E. Houstis and J. Rice. Computer as Thinker/Doer: Problem-Solving Environments for Computational Science, IEEE Computational Science and Engineering, summer 1994.
- [3] John R. Rice and Ronald F. Boisvert, From Scientific Software Libraries to Problem Solving Environments, IEEE Computational Science and Engineering Magazine, fall 1996, pp. 44-53.
- [4] Balay R.I, Vouk M.A., Perros H., "Performance of Network-Based Problem-Solving Environments," Chapter 18, in Enabling Technologies for Computational Science Frameworks, Middleware and Environments, editors Elias N. Houstis, John R. Rice, Efstratios Gallopoulos, Randall Bramley, Hardbound, ISBN 0-7923-7809-1, 2000
- [5] Ken Brodlie, Andrew Poon, Helen Wright, Lesley Brankin, Greg Banecki and Alan Gay, GRASPARC – A Problem Solving Environment Integrating Computation and Visualization. IEEE Conference, 1993.
- [6] Nathan A. DeBardeleben, Walter B. Ligon, Sourabh Pandit and Dan C. Stanzione Jr. Coven: a Framework for High Performance Problem Solving Environments, IEEE International Symposium on High Performance Distributed Computing HPDC-11 2002.
- [7] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Satish Thatte, Dave Winer and Henrik Nielsen. “Simple Object Access Protocol (SOAP)” 08-May-2000. <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

- [8] Universal Description, Discovery and Integration, Executive White Paper, 14-Nov-2001, [http://www.uddi.org/pubs/UDDI\\_Executive\\_White\\_Paper.pdf](http://www.uddi.org/pubs/UDDI_Executive_White_Paper.pdf)
- [9] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana “ Web Services Description Language (WSDL)”, 15-Mar-2001, <http://www.w3.org/TR/wsdl>
- [10] Tim Bray, Jen Paoli, C. M. Sperberg-McQueen, Eve Malor “Extensible Markup Language (XML) 1.0, 6-Oct-2000, <http://www.w3.org/TR/REC-xml>
- [11] Singh M.P., Vouk M.A., "Scientific workflows: scientific computing meets transactional workflows," Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions, Univ. Georgia, Athens, GA, USA, 1996.
- [12] Christophe Coenraets, Web Services: Building the next generation of E-Business applications, 2-Oct-2001.  
<http://www.javaworld.com/white-paper/macromedia-webservices-021202.pdf>
- [13] Doug Tidwell, Web Services: The Web’s next Revolution.  
<https://www6.software.ibm.com/developerworks/education/wsbasics/wsbasics-a4.pdf>
- [14] Qusay H. Mahmoud. Registration and Discovery of Web Services, June 2002.  
<http://developer.java.sun.com/developer/technicalArticles/WebServices/jaxrws/>
- [15] Axis, A SOAP implementation, <http://xml.apache.org/axis/>
- [16] IBM WSTK 2.3 <http://www.alphaworks.ibm.com/tech/webservicestoolkit/>
- [17] Apache Tomcat <http://jakarta.apache.org/tomcat/tomcat-4.0-doc/index.html>
- [18] WebSphere Studio Application Developer Integration Edition 4.0, Services: Definition, Binding, Invocation, IBM Services Group.  
[http://www7b.software.ibm.com/wsdd/library/presents/AppDevIE\\_Training.html](http://www7b.software.ibm.com/wsdd/library/presents/AppDevIE_Training.html)

- [19] Cluster and Factor Analysis with Varimax Orthogonal Rotation: CLUSFAVOR  
<http://mbcr.bcm.tmc.edu/genepi/> 2002.
- [20] National Center for Biotechnology Information: NCBI GenBank  
<http://www.ncbi.nlm.nih.gov/Genbank/index.html> 2002.
- [21] Basic Local Alignment Search Tool: BLAST Basic Overview  
[http://www.ncbi.nlm.nih.gov/BLAST/blast\\_overview.html](http://www.ncbi.nlm.nih.gov/BLAST/blast_overview.html) 2002.
- [22] TRANSFAC - The Transcription Factor Database  
<http://transfac.gbf.de/TRANSFAC/>
- [23] Ling Liu, Calton Pu, Wei Han, David Buttler, Henrique Paques, Wei Tang. XWRAP  
Java Wrappers <http://www.cc.gatech.edu/projects/disl/XWRAPelite>
- [24] “What is bioinformatics” <http://www.ncbi.nlm.nih.gov/Education/>
- [25] Volker Turau, Transforming relational databases into XML documents. DB2XML  
1.4, <http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/>
- [26] Thomas E. Potok, Mark Elmore, Joel Reed, Nagiza Samatova and Nenad Ivezic.  
“VIPAR: Advanced Information Agents discovering knowledge in an open and changing  
environment”.
- [27] O. Lassila, R. Swick. “Resource Description Framework (RDF) Model and Syntax  
Specification”, W3C Recommendation, 22-Feb-1999, <http://www.w3.org/RDF/>
- [28] Thomas E. Potok, Nagiza Samatova and Nenad Ivezic. An Ontology based HTML to  
XML Conversion Using Intelligent Agents. 35<sup>th</sup> Annual Hawaii International Conference  
on System Sciences (HICSS '02).

- [29] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, I. Horrocks, The Semantic Web: The roles of XML and RDF, IEEE Internet Computing, Vol. 4(5), Sept/Oct 2000.
- [30] Ranieri Baraglia, Renato Ferrini, Domenico Laforenza, Meta: A Web-based Metacomputing Environment to build a Computational Chemistry Problem Solving Environment, Parallel, Distributed and Network-based Processing, 2002.
- [31] Gregor von Laszewski, Ian Foster, Jarek Gawor, Peter Lane, Nell Rehn, Mike Russell. Designing Grid Based Problem Solving Environments and Portals, System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference, 2001
- [32] I-Flow Architecture White Paper  
[http://www.i-flow.com/about\\_iflow/white\\_papers/InterstageiFlowWP.pdf](http://www.i-flow.com/about_iflow/white_papers/InterstageiFlowWP.pdf)
- [33] TurboBench, [http://www.turbogenomics.com/products/turbobench\\_overview.html](http://www.turbogenomics.com/products/turbobench_overview.html)
- [34] VIBE: Visual Integrated Bioinformatics Environment, Incogen Inc.  
[http://www.incogen.com/vibe/VIBE\\_WhitePaper.pdf](http://www.incogen.com/vibe/VIBE_WhitePaper.pdf)
- [35] Jason Z.Xiang, BINGO “Graphical Environment for Sequence Analysis and Structure Modeling” <http://trantor.bioc.columbia.edu/~xiang/bingo/>
- [36] M. Weske, G. Vossen: The WASA Project: A Survey; in presented at 1st European Workshop on Workflow and Process Management 1998, Zurich, October 1998.
- [37] OMG, The Common Object Request Broker. <http://www.omg.org/>
- [38] David Hall and John Miller, GeneFlow <http://gene.genetics.uga.edu/workflow/>
- [39] Krys Kochut, John Miller, Amit Seth. Meteor (Managing End-to-End OpeRations)  
<http://lsdis.cs.uga.edu/proj/meteor/meteor.html>

- [40] Jeffrey L. Brown, Thomas C. Hudson, Ann E. Stapleton, Jim Blum, Sridhar Narayan, Gene Tagliarini, Ron Vetter and Kenisha V. Johnson. GeneBeans, <http://people.uncw.edu/hudson/publications/ismb02.html>
- [41] Lightweight Directory Access Protocol (LDAP) <http://www.openldap.org/>
- [42] Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 2001, [www.globus.org/research/papers/anatomy.pdf](http://www.globus.org/research/papers/anatomy.pdf)
- [43] Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke “The Physiology of the Grid – An Open Grid Services Architecture for Distributed Systems Integration”
- [44] James Snell “Introducing the Web Services Flow Language”.
- [45] Arindam Banerji, Claudio Bartolini, Dorothea Beringer, Venkatesh Chopella, Kannan Govindarajan, Alan Karp, Haruni Kuno, Mike Lemon, Gregory Pogossiants, Shamik Sharma, Scott Williams, WSCL 1.0, 14-Mar 2002.
- [46] Empowering small and medium sized businesses to compete in the e-Marketplace [http://www.internosis.com/aboutus/News/articles/bts\\_white\\_paper.pdf](http://www.internosis.com/aboutus/News/articles/bts_white_paper.pdf)
- [47] Satish Thatte “XLANG- Web Services for Business Process Design”, 2001
- [48] IBM International Technical Support Organization (IBM Redbooks) Workflow and Image Library: FlowMark and VisualInfo with Windows. IBM 08/96
- [49] The Open Bioinformatics Foundation <http://www.open-bio.org/>
- [50] Lincoln Stein “Creating a Bioinformatics Nation” <http://bio.oreilly.com/>
- [51] Discovery Link - IBM’s Life Sciences Framework, 2002 <http://www-3.ibm.com/solutions/lifesciences/solutions/discoverylink.html>
- [52] Scientific Data Management Center, <http://sdm.lbl.gov/sdmcenter/>

- [53] Vouk, M. A., R. Balay, and J. Ambrosiano, 1995: EDSS - An Environment for Large-Scale Numerical Computing and Decision Making. International IFIP/WG 2.5 Workshop on Current Directions in Numerical Software and High Performance Computing, Kyoto, Japan, October 16-17.
- [54] Lurch M, Kafura D, Symphony – a Java based Composition and Manipulation Framework for Computational Cluster. Computing and the Grid 2nd IEEE/ACM International Symposium CCGRID2002, 2002
- [55] Balay R., M. A. Vouk, H. Perros " A Lightweight Software Bus for Prototyping Problem Solving Environments", Systems Engineering Conference, Las Vegas, 1996.
- [56] Designing grid-based problem solving environments and portals Von Laszewski G., Foster I., Gawor J., Lane P., Rehn, N., Russell M., System Sciences, 2001. 34th Annual Hawaii International Conference on, 2001
- [57] "Activity Networks: Project Planning & Control by Network Models," by Salah E. Elmaghraby, November 1977, Wiley, John & Sons, Inc., and Elmaghraby S.E., "On generalized activity networks," J. Ind. Eng., Vol. 17, 621-631, 1966]
- [58] Francisco Curbera, Goland, Klein, Leymann, Roller, Weerawarana and Thatte Business Process Execution Language for Web Services, 31 July 2002  
<http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>
- [59] OSWorkflow – OpenSymphony, <http://www.opensymphony.com/osworkflow/>
- [60] OFBiz – The Open for Business Project, <http://www.ofbiz.org/>
- [61] OBE – Open Business Engine, <http://www.openbusinessengine.org/index.html>
- [62] XML Process Definition Language (XPDL), Workflow Management Coalition Specification May 22, 2001, [http://www.wfmc.org/standards/docs/xpdl\\_010522..pdf](http://www.wfmc.org/standards/docs/xpdl_010522..pdf)

## Appendix A: List of Tools Surveyed

1. WebSphere Application Development Integration Edition 4.0: WSAD IE is a commercial product from IBM. Services: Definition, Binding, Invocation, IBM Services Group.  
[http://www7b.software.ibm.com/wsdd/library/presents/AppDevIE\\_Training.html](http://www7b.software.ibm.com/wsdd/library/presents/AppDevIE_Training.html)
2. i-Flow, Fujitsu Software Corporation <http://www.i-flow.com/index.htm>
3. BizTalk, Microsoft Corporation <http://www.microsoft.com/biztalk/>
4. BEA WebLogic Collaborate 2.0  
[http://edocs.bea.com/wlintegration/v2\\_0/collaborate/workflow/index.htm](http://edocs.bea.com/wlintegration/v2_0/collaborate/workflow/index.htm)
5. TurboBench [http://www.turbogenomics.com/products/turbobench\\_overview.html](http://www.turbogenomics.com/products/turbobench_overview.html)
6. VIBE: Visual Integrated Bioinformatics Environment, Incogen Inc.  
<http://www.incogen.com>
7. Jason Z.Xiang, BINGO “Graphical Environment for Sequence Analysis and Structure Modeling” <http://trantor.bioc.columbia.edu/~xiang/bingo/>
8. The WASA Project: Flexible Workflow Management  
<http://dbms.uni-muenster.de/menu.php3?item=projects>
9. GeneFlow <http://gene.genetics.uga.edu/workflow/>
10. Meteor (Managing End-to-End Operations). Krys Kochut, John Miller, Amit Seth. <http://lsdis.cs.uga.edu/proj/meteor/meteor.html>
11. GeneBeans: a bioinformatics workflow and data management system  
<http://people.uncw.edu/hudson/publications/ismb02.html>



12. WIDE: Workflow on Intelligent Distributed database Environment  
<http://dis.sema.es/projects/WIDE/>
13. Panta Rhei: Workflow Research at the University Klagenfurt  
<http://www.ifi.uni-klu.ac.at/ISYS/JE/Projects/Workflow/>
14. Tom Sawyer Software Graph Editor Toolkit  
<http://www.tomsawyer.com/get/get-java.html>
15. Graphviz – Open source graph drawing software  
<http://www.research.att.com/sw/tools/graphviz/>
16. Flow ware: Plexus <http://www.plx.com/>
17. BioExchange: Enabling the Life Sciences  
<http://www.bioexchange.com/tools/software.cfm?start=31>
18. WEAVE: Workflow to Enable Agile Virtual Environment  
<http://www.cs.sunysb.edu/~workflow/>
19. Crossflow: Cross Organizational Workflow  
<http://www.darmstadt.gmd.de/oasys/projects/crossflow/index.html>
20. Active Workflow: Singularity  
[http://www.singularity.co.uk/Our\\_Products/Our\\_Products.htm](http://www.singularity.co.uk/Our_Products/Our_Products.htm)
21. Workflow: eiStream Enterprise Workflow  
<http://www.eastmansoftware.com/products/>
22. Optix, Document Management and Workflow Systems
23. OSWorkflow <http://www.opensymphony.com/osworkflow/>
24. OFBiz (Open for Business) <http://www.ofbiz.org/>
25. OBE (Open Business Engine) <http://www.openbusinessengine.org/index.html>