Extending the SOA paradigm to e-Science environments

Andrea Bosin[1,2], Nicoletta Dessì[2], Barbara Pes[2]
[1]Istituto Nazionale di Fisica Nucleare (INFN), Sezione di Cagliari, I-09042 Monserrato, Italy
[2]Università degli Studi di Cagliari, Dipartimento di Matematica e Informatica, Via Ospedale 72, 09124 Cagliari, Italy
E-mail: {andrea.bosin}@dsf.unica.it, {dessi,pes}@unica.it

Pre-print submitted to Future Generation Computer Systems (Elsevier)

Abstract

In the business world, Service Oriented Architecture (SOA) has recently gained popularity as a new approach to integrate business applications. In a similar way, scientific workflows can accomplish the automation of large-scale e-Science applications. However, the use of workflows in scientific environments differs from that in business environments. Scientific workflows need to handle large amounts of data, deal with heterogeneous and distributed resources such as the Grid, and require specialized software applications that are written in diverse programming languages, most of which are not popular in business environments.

In this paper, we analyze the preparedness and the shortcomings of the SOA paradigm in addressing the needs of e-Science and the extent to which this can be done. The paper identifies the characteristics of a Virtual Organization providing scientific services, and presents a model placing particular emphasis on BPEL processes as a mean for supporting the interaction with Web Services. We discuss the challenges encountered in the seamless integration of BPEL processes within an e-Science infrastructure and we propose a set of complementary infrastructural services. By providing business utilities and automation technology founded onWebServices, infrastructural services cooperate with BPEL in ensuring on-demand resource provisioning for the execution of scientific workflows, while addressing some critical issues such as security, access control and monitoring. Furthermore, the paper presents our experience in adopting the proposed approach within a collaborative environment for bioinformatics. To illustrate how a scientific experiment can be formalized and executed, we focus on micro-array data processing, a field that will be increasingly common in applications of machine learning to molecular biology.

Keywords: E-Science, Distributed systems, Workflow management, Web Services

1. Introduction

Recent advances in computer science facilitate a paradigmatic shift in the way researchers conduct modern science, and enable scientists to validate new research practices for many scientificapplications such as biology, astronomy and particle physics.

Scientists of today routinely rely on computers and data and information sharing over the Internet to aid them in their research activities. Often, scientific progress is achievable only through large-scale international collaborations: examples are the human genome project and particle physics experiments. The term e- Science [1] refers to this type of large-scale cooperation, within a typical experimental scenario that requires data to undergo several pre-processing, computing, post-processing and output storage stages.

Workflow systems [2] provide specialized computing environments for automating this process, allowing the scientist to orchestrate a set of abstract tasks that represent the experimental stages, while hiding the complexity associated with programming languages, libraries and hardware infrastructures.

Many scientific workflow systems are currently in use [3]. However, each system only tailors some (specific) technical aspect such as the use of computational resources [4-7], user-friendly interfaces [8-10] or applications related to a particular scientific domain [11-13]. Often, the monolithic design strategy of scientific workflow systems has hindered the development of a common terminology and standards, resulting in systems that support solutions that are only sui able for the specific problem at hand and for a specific execution environment.

A similar situation occurs in industrial and business domains where a Service Oriented Architecture [14,15] (SOA) is advocated for improving interoperability between systems engaged in mutually dependent business activities. In the SOA paradigm, functionality provided by business applications is enclosed within Web Service (WS) [16] technology, and this has stimulated a growing interest in research environments. Exposed as Web Services, scientific applications can be integrated into complex workflows, which may be deployed on multiple and distributed resources. However, since Web Services do not preclude the use of other technologies (for example middleware solutions), the point is not whether the WS technology can be adopted by e-Science environments, but if it can be done in such a way to access and virtually integrate a potentially enormous number of physically distributed resources, belonging to different organizations that may use a wide variety of mechanisms (schedulers, reservation systems, control interfaces, authentication, etc.).

Recent work [17,18] has shown that embracing BPEL for building and enacting service-based workflows in e-Science contexts still remains a challenging task [3,19], especially when the scientific environment is composed by heterogeneous and distributed resources whose status (e.g., CPU or storage availability) changes dynamically [20].

In this paper, we analyze the preparedness and the shortcomings of the SOA paradigm in addressing the needs of e-Science environments and propose an approach for on-demand resource provisioning in distributed heterogeneous environments, including the Grid.

Our proposal borrows many SOA concepts and standards from the business domain, including the adoption of BPEL. A motivation of our approach is almost evident: the SOA paradigm, and in particular Web Services and BPEL, are based on widely accepted standards and are supported by many software tools, both open source and commercial.

Our goal is not to show that SOA is the best way for implementing scientific workflows, but rather to suggest a new direction, moving workflow systems from the class of specialized applications towards the class of open standards service-based systems for e-Science environments. This will allow scientific workflow systems to achieve their potential as a new paradigm for supporting the science socialization invoked as the next important step in e-Science environments [12].

Moreover, the paper aims at:

- presenting an in-depth understanding of the fundamental issues underpinning the application of SOA for e-Science purposes;
- exposing the necessary background for addressing the challenges associated with the establishment of an e-Science environment; and
- suggesting a way for the exploitation of the SOA approach in the e-Science domain.

The paper is organized as follows. First, we introduce the basic characteristics of a scientific Virtual Organization (VO), whose participants are characterized by the common purpose of executing scientific experiments through the design and enactment of workflows. Then, we analyze the major limitations regarding the integration of SOA and the Grid, and the execution of BPEL workflows. After that, we discuss an on-demand resource provisioning system that can interface a heterogeneous distributed computing infrastructure including the Grid. Finally, a reference implementation is presented and validated in the scenario of a bioinformatics experiment designed and executed through a BPEL workflow.

2. Extending the SOA paradigm to scientific virtual organizations

In this section we model a collaborative environment for a scientific VO, i.e., a coordinated group of organizations (institutions and/or individuals) who collaborate towards a common scientific goal and share, on the basis of some policies, a set of resources (computing, storage, instruments and data) and services.

Organizations offer services to their *users* (researchers that consume services), and consume services provided by other organizations. The collaboration between organizations promotes the development and deployment of value-added services that users throughout the whole VO can consume. Organizations manage *resources* (computing, storage, instruments and data which live encapsulated within the organizations and support them in accomplishing their goals), *infrastructural components* (architectural components embedding the technology that supports collaboration), *business components* (software components providing scientific applications) and *business processes* (workflows expressing scientific experiments).

Every organization belonging to a scientific VO has to face the problem of interoperating a number of software components:

- Workflows define and coordinate complex scientific experiments in terms of business processes.
- Service interfaces expose the business logic of scientific applications. They support communication contracts (messagebased communication, formats, protocols, security, etc.) that different consumers require.
- Components implement business rules and perform business tasks related to a specific scientific domain, regardless of whether a scientific process consists of a single service or an orchestrated workflow.
- Data access components provide a standard and abstract way for accessing the storage systems as required by most services at some point during a scientific process.
- Ports are access points to services as they offer a general schema for the publish-and-subscribe pattern.
- Data structures wrap datasets, data-readers or XML streams that are used internally for data exchange between components.

Each partner in a VO has the same abstract organization shown in Fig. 1 and described below.

The bottom layer, namely the fourth, represents various computing, storage and instrumentation facilities that address a large spectrum of resources. At one end of the computing spectrum are desktop/laptop hardware and software for simple data analysis and visualization. At the other end we find resources organized in clusters managed by some kind of lower-level scheduler (e.g. LSF [21], SGE [22]), possibly offering the image of a single system through the use of Grid middleware such as gLite [23] or Globus [24]. Legacy tools and applications, database systems and interfaces to devices are also found at this level.

One level higher, the third layer consists of the interfaces and software assets to computing, storage and instrumentation. It is provided by services and components that manage and allow the access to heterogeneous, distributed resources and widely deployed infrastructures. The services at this level abstract the access to resources in such a way to almost completely decouple it from the low-level details (hardware, operating systems, schedulers, etc.).

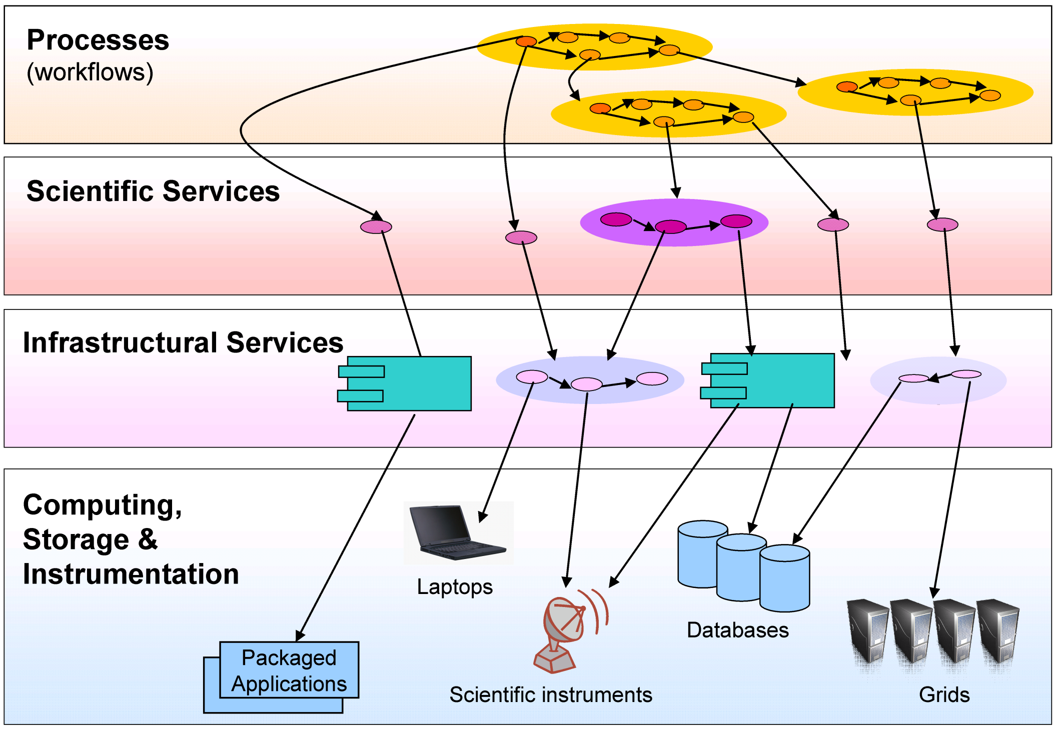

Fig. 1. The abstract organization of a VO partner.

This layer is enabled by:

- Infrastructure services built on various middleware components and APIs, etc.
- Grid collective services supporting standards like the Job Submission Description Language (JSDL) or OGSA-DAI (Open Grid Service Architecture Data Access and Integration) services.

The second layer includes scientific business services that can be invoked as standalone activities or can be orchestrated by a workflow engine. Such services are fine-grained, loosely coupled and self-contained and automate experimental tasks, provide value to the collaborative environment and are part of standard experiment processes. According to the technology, we can find

- Web Services natively developed from WSDL documents, and
- Web Services that wrap existing software applications (e.g., scientific libraries).

Finally, the workflow layer manifests the business processes related to scientific experiments and consists of BPEL workflows that orchestrate the services belonging to the second and third levels.

Among the technologies enabling the above scenario, the Grid takes a leading role since it addresses issues related to access provisioning, coordinated resource sharing, usage and security policies, etc. Moreover, Open Grid Service Architecture (OGSA) defines an open and extensible framework for merging Web Services and Grid technologies.

The VO we consider is a Grid-aware cooperative system, where organizations live on the Internet and offer their services as Web Services described by WSDL standards.

3. Integration issues

The integration of the different technologies belonging to each layer of the collaborative environment presented in the previous section raises a number of problems. Some of these are discussed in the following subsections.

3.1. Resource management

Scientific services are executed on resources, including the Grid, which are heterogeneous and geographically distributed. In addition, a scientific VO may comprise many different institutions and thousand of researchers that collectively control hundred or thousands of computers. Each individual researcher and institution may participate in, and contribute resources to, multiple collaborative projects that can vary widely in scale, lifetime and application domain. A standard way to manage such resources in order to allow a researcher to use only one mechanism to request and use these resources is then mandatory.

OGSA services provided by Grid environments, according to OGSA WSRF Basic Profile 1.0 [25], are based on the WS-RF [26] specifications which are concerned with the creation, addressing, inspection and lifetime management of stateful abstract resources, namely WS-Resources. The properties of a WS-Resource are described by a resource property document and addressed by means of endpoint references (EPRs). However, a general scientific VO is composed not only of Grid resources, but also of other heterogeneous and distributed resources that are managed by a wide variety of mechanisms (schedulers, queuing systems, etc.). In addition, many existing Web Services developed according to the WS-I Basic Profile 1.1 [27] are not concerned with WS-Resources, and most of the applications running on the Grid are not even Web Services.

3.2. Scientific workflow features

Scientific workflows have significant departures from business workflows.

The primary difference stems from the fact that business workflows are built on a set of fixed repetitive tasks, while a scientist is systematically altering the data and the execution patterns in the workflow looking for new interesting or unexpected outcomes. This means that a single workflow often requires many applications to be executed. Such applications are built with many different technologies; i.e., they can be provided as libraries or binaries, delivered as Web Services or web portals. Moreover, these applications are heterogeneous in their logic (since they can fit a variety of domains) and in their execution, all dealing with different requirements (i.e., high-performance execution, graphic facilities, fast database access, data streaming, etc.).

The second difference featuring scientific workflows is in the need for processing large amounts of data from many sources to gain the information needed during the experiment or to compare results with. Thanks to the global network, the amount of data available in public and specialized databases is rising exponentially. Data can be accessed in a variety of a ways; often it is possible to download data directly from public repositories, while access to databases requires proper wrappers. Additional problems occur in pipelining, i.e., in passing data between applications, because this requires managing input and output formats.

Finally, one of the foundations of modern science is that experiments must be repeatable. This suggests that the tasks making up the scientific workflows should be stored and reused in other experiments. After successfully designing and executing a workflow, a scientist would like to save the workflow description and to publish it, so all the members of the VO can execute the same workflow with their own data. This aspect is very important for scientific reproducibility.

3.3. Integrating services in BPEL processes

As with a business application, a scientific workflow is a whole that expresses the experiment strategy, and BPEL offers a standardized way to describe its functional composition. However, BPEL has some shortcomings that limit its usability in scientific VOs. These shortcomings do not originate from the BPEL language “per se”, but from some weaknesses exposed by BPEL in workflow execution.

One drawback concerns the potential offered by BPEL in procuring resources on heterogeneous distributed environments, including the Grid. Dynamic resource provisioning is not part of the language, but it is an essential feature of the reference model proposed in Fig. 1, where the resources belonging to the fourth layer may change during workflow execution, or may not even exist before they are requested and created.

In BPEL, partner links are references to the Web Service interfaces participating in the process and they contain the endpoint reference of the services. Such endpoints are usually static and come from the WSDL documents describing the services, but they can be assigned dynamically during process execution, thus allowing for a true dynamic provisioning of service implementations.

Unfortunately, BPEL does not provide a simple way to work with WS-RF services and resources. Even if is possible to manually handle SOAP headers within a BPEL process, this is cumbersome and error prone. Direct interaction with OGSA-compliant Grid services is then a problem.

Moreover, available BPEL workflow engines are not fully interoperable with the security features of existing Grid middleware. Such features are, for example, the support of the Globus toolkit infrastructure that relies on proxy certificates, or the support of SAML assertions to present additional signed security tokens.

Also, the long-running nature of scientific workflows imposes the availability of mechanisms for progress monitoring, inspection of intermediate results, and error handling at runtime.

Error and compensation handlers offered by BPEL operate at design time and do not provide appropriate support to workflow monitoring. This makes it difficult to control the irregular behaviours related to infrastructural failures such as network timeouts and software faults in invoked services. Consequently, compensation mechanisms for workflow fault-tolerant execution are regarded as desirable.

The second drawback is related to the data-intensive nature of many scientific experiments. Input data must be transferred to the data-processing services, while output data must be transferred back from the same services. BPEL follows the concept of centralized control and centralized data flow. That means that the BPEL engine is the broker for all message exchanges between all the Web Services participating in the workflow. Embedding huge data sets in SOAP messages is possible, but this is not a suitable solution from the point of view of performance, and may lead to serious problems of scalability of the BPEL engine. On the contrary, data transfers should be minimized and storage resources for parking temporary data should be available in the neighbourhood of the computing resources needing them.

4. Infrastructural services for on-demand resource provisioning

4.1. Introduction

The above issues prevent the straightforward application of the SOA paradigm within the proposed scientific VO. As a possible solution, we propose a set of Web Services, namely infrastructural services, which implement a dynamic resource provisioning system for the execution of scientific business services orchestrated by a standard BPEL workflow.

Both the BPEL process and the infrastructural services complement each other in workflow execution, but we prefer to keep a clear separation of concerns between them, in such a way as to decouple BPEL workflow and resource management.

In more detail, the infrastructural services are responsible for

- the computing resources' life cycle: allocation, activation, monitoring, de-allocation and accounting;
- access to computing resources; and
- access to (temporary) storage resources needed by scientific services during workflow execution.

The BPEL process is responsible for

- obtaining resources from the infrastructural services and using them for the execution of scientific business services; and
- managing the control and data flow of the scientific business services.

In designing infrastructural services we adopt a decentralized approach that categorizes services as global and local.

In more detail, *global services* are the only components that interact with the workflow engine. They are included into a BPEL process as partner links. Global services carry on the resource procurement and provide all the functionalities for user registration and authentication. In addition, they maintain information about the resources provided and have the capability to monitor the resources in order to obtain up-to-date information about their status.

Local services are responsible for managing the resources of specific types (e.g., Grid-aware resources), needed for the execution of the scientific services. They can be invoked exclusively by the global services and act as their intermediaries in interfacing with the lower-level middleware.

It is worth noticing that a BPEL process is a Web Service itself. We have then a hierarchy of services (BPEL process, global and local services) in which a global service may have to act as a local service for the global BPEL workflow and can be executed by a BPEL engine. This enables one to recursively partition Web Services into aggregations of subservices, hiding the details of the aggregation to higher-level services that BPEL orchestrates. This in accordance with the SOA privacy principles which allow designing services at the enterprise level (based on intra-company services), and then exposing such services in a public repository.

In this sense, global services can be considered as public services. They are accessible, in a standard way, by users and can be orchestrated by BPEL. In contrast, local services are private services since they can be invoked exclusively by the global services that are responsible for their coordination.

A distinctive aspect of the proposed solution is that it operates as a bridge that connects the two basic components of the SOA architecture, the service client (i.e., the BPEL instance) and the scientific business service providers, by means of the infrastructural services. As scientific services are needed, the BPEL engine invokes the infrastructural global services - which are organized to coordinate an integrated and self-contained set of local services - for procuring the resources needed for their execution.

4.2. Service definition

The most important global and local services are shown in Fig. 2 together with the other parties, BPEL processes and resources, and are described below.

The *Resource Allocation* (RA) global service implements the public interface for resource provision and management. It acceptsuser requests for resource allocation/release, notifies users when resources are available, keeps track of the resource endpoint and status, and coordinates other infrastructural services. It provides logging and monitoring capabilities for all the resources provisioned to workflows.

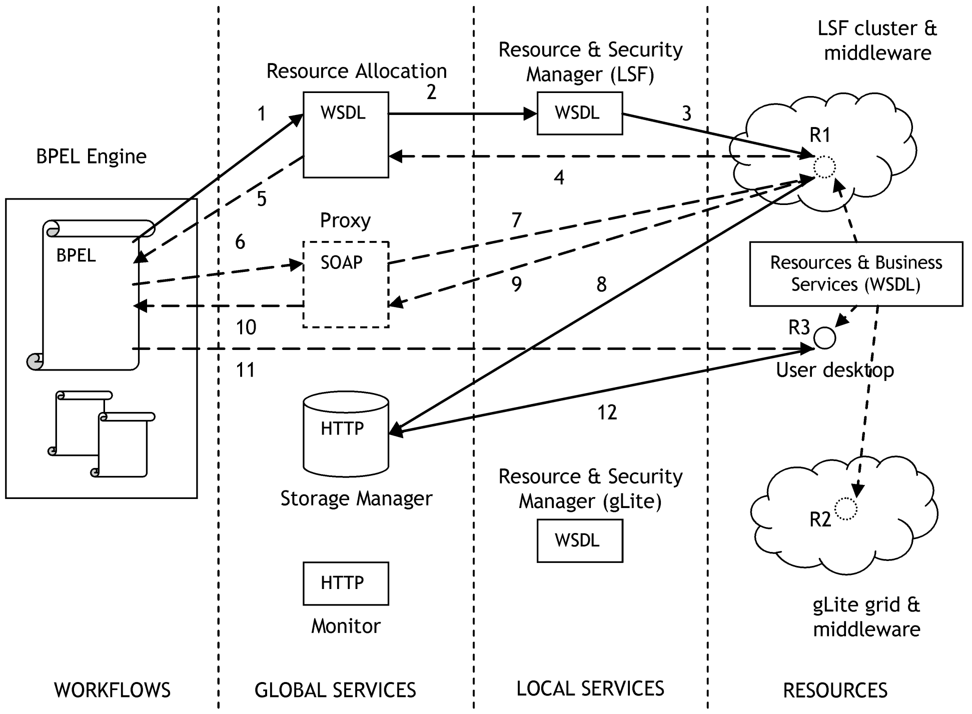

Fig. 2. Global and local services.

The *Proxy* global service acts as an intermediary for routing SOAP messages from BPEL to scientific business services and vice versa. Its Business Service Proxy (BSP) component accepts all user requests directed to scientific services and routes them to the service endpoint (which may be on a private network); in the case of request-response messages it can route the service response back to the user. The Notification Proxy (NP) component is responsible for routing the notification messages generated by the scientific services to the user.

The *Storage Manager* (StM) global service provides temporary storage to data-intensive scientific business services. When the output from a scientific service serves as input for a subsequent service invocation, it can be saved instead of transferring it back and forth between BPEL processes. The Storage Manager assigns a public URL to the data so that it can be easily retrieved when needed. Many StMs can coexist, thus ensuring optimal performance and scalability.

The *Monitor* global service is a simple service that shows information about the resource context related to scientific services, such as resource identifier, status, WSDL port type or last operation invoked on the scientific service.

Resource allocation and user authentication are delegated to the specific local services.

The *Resource Manager* (RM) local services are wrappers over the underlying middleware or legacy schedulers. Their role is to accept requests from the RA, verify user credentials by invoking the Security Manager, and dispatch them to the underlying middleware/scheduler.

The *Security Manager* (SM) is a local service that addresses security and access control requirements according to the policies enforced on the resources by the individual organizations belonging to the VO. User authentication mechanisms are often resource specific, and can range from no authentication for publicly available resources to proxy certificates for Grid users or to digitally signed security tokens for LDAP users with a public key certificate. If the authentication is successful, the Security Manager issues an authorization token (e.g., a session password) that grants access to the given resource.

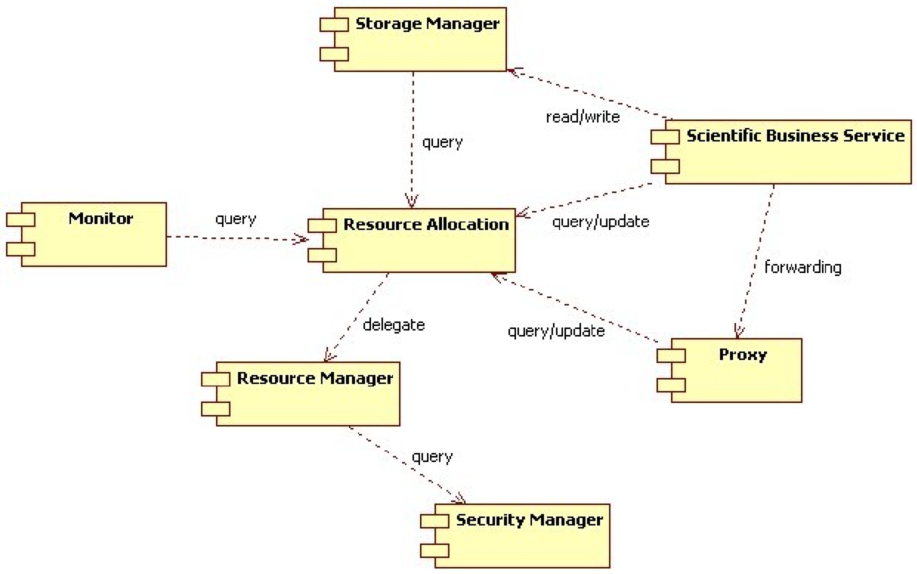

Fig. 3. Dependencies between infrastructural services.

The component diagram in Fig. 3 shows the dependencies between the various infrastructural services.

4.3. Service interaction

To show the role of infrastructural services in the execution of a scientific workflow, let us consider a sample collaborative scenario in which a researcher belonging to a scientific VO wishes to execute a scientific application and view the result. The VO provides a number of computing resources, shown in the right part of Fig. 2, and grants to the researcher access to a cluster managed by some batch scheduler (LSF in Fig. 2). Both the scientific application and the viewer are Web Services, and the former needs a suitable computing resource to be executed, while the latter runs on the user desktop. The flow of execution is described by a BPEL process, and the researcher enacts it by means of a BPEL engine (which could also be running on the user desktop).

The arrows numbered from 1 to 12 in Fig. 2 show a typical collaboration sequence between the BPEL process, infrastructural services and resources. Arrows with dashed lines represent a one-way-interaction while continuous lines represent request-response invocations.

1. The Resource Allocation global service is invoked by the BPEL process and it is in charge of provisioning a suitable computing resource to execute the given scientific service; the resource will be allocated on the LSF cluster for which the researcher has provided authentication credentials; the BPEL process itself is set as the endpoint for all asynchronous notifications; if the invocation is successful (i.e., steps 2 and 3 are executed without errors) the resource identifier is returned.
2. The Resource Manager and Security Manager local services responsible for the LSF cluster are invoked by the Resource Allocation global service; the authentication credentials are verified.
3. If the credentials are valid, the LSF scheduler is directed to allocate a resource (R1 in Fig. 2) and, when available, to start on it the given scientific service (the service binary is downloaded on-the-fly from a network URL).

4. When a resource is available and the scientific service has been started, an asynchronous notification is sent to the Resource Allocation service; the notification message contains all the information about the resource context (resource identifier, private endpoint, etc.); a mapping between the resource identifier and its context is then created.
5. An asynchronous notification is sent to the BPEL process; the notification message contains the resource identifier and the public endpoint reference of the scientific service, which points to the Proxy global service.
6. The BPEL process assigns the public endpoint reference of the scientific service to the corresponding partner link and invokes it with the input message; the public endpoint points to the Proxy, so the message is sent to the Proxy; the input message contains the resource identifier.
7. The Proxy uses the resource identifier as an index and retrieves the private endpoint of the scientific service from the resource context; then it forwards the input message to the scientific service.
8. The scientific service performs its computation on the data contained in the input message (the input message may also indicate that data must be retrieved from a network URL, e.g., from the Storage Manager); then it sends the result to the Storage Manager global service for temporary storage; the Storage Manager returns a public URL for retrieving the saved data.
9. The scientific service sends an asynchronous notification to the Proxy; the notification message contains the resource identifier and the result in the form of a public URL.
10. The Proxy Service notifies the BPEL process.
11. The BPEL process invokes the viewer service on the user desktop (R3 in Fig. 2) with an input message containing the public URL of the result to be viewed.
12. The viewer retrieves the result previously saved (step 8), from the Storage Manager public URL, and shows it to the researcher.

In the sample scenario presented above, and in the case study/experiment presented in Section 6, resource allocation is done using real-life authentication such as Grid proxy certificates or digital signature, but the access to scientific services is granted simply by checking the resource identifier contained in the input messages. In the general case, where a stronger authentication is needed, an authorization token (e.g., a session password) issued by the Security Manager should also be used and HTTPS protocol should be used to transport SOAP messages (as with Grid Services). In a prototype environment this is an additional complication, and we have not yet implemented it.

5. Implementation details

In this section, we present an overview of the prototype implementation that aims to provide seamless access to thedifferent resources available through our local VO infrastructure, e.g.,

- Clusters of Linux servers managed by LSF and SGE.
- Access to a nation-wide gLite-based Grid infrastructure (IGI, Italian Grid Infrastructure [28]).
- Standalone servers and desktop computers.

The programming framework we have chosen is Java 1.6, thanks to its support to Web Services, servlets, SOAP, HTTP, gLite and security. The requirement of the availability of Java Runtime Environment (JRE) for the execution of scientific services is not a problem, since JRE can be installed even on-the-fly in a download-install-run-uninstall fashion.

As already discussed, in the global services we have avoided sophisticated WS-* protocols, such as WS-RF or WS-Security, which are difficult to use in conjunction with BPEL and need extra Java libraries. WS-Addressing has been employed for the management of dynamic endpoint references of scientific business services. All the information exchanged through SOAP messages has been placed into the body of the messages.

5.1. Resource description

In our vocabulary, a resource is a logical entity which spans a subset of the computing and storage elements available to the VO, and has its own context, e.g., the resource properties document of a WS-Resource. The complete context is in fact unnecessary to the BPEL process provided it knows:

- the resource identifier to establish correlations;
- the authorization token that grants access to the resource; and
- the resource endpoint reference to perform dynamic assignment to the corresponding partner link.

In contrast, infrastructural services need the complete context of each resource to perform the required operations (query, release, etc.), but this can be managed by keeping a mapping between resource context and identifier.

To perform a scientific experiment described by a BPEL process, we usually need one resource for each scientific service, namely a resource set. A resource set request specifies the endpoint (usually the BPEL process) to use for the asynchronous notification of resource availability, and the set of the specific resources. An XML fragment describing a resource set request is shown in Fig. 4.

```
<resourceRequest>
  <notificationEndpoint>
    <add:EndpointReference
      xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
      <add:Address>
        http://10.12.0.6:8100/ResourceAllocatorNotification
      </add:Address>
    </add:EndpointReference>
  </notificationEndpoint>
  <BSRequest>
  ...
  </BSRequest>
  <BSRequest>
  ...
  </BSRequest>
</resourceRequest>
```

Fig. 4. The XML fragment describing a resource set.

For each resource in the resource set, the request specifies the resource characteristics and the specific scientific business service to be executed:

- scheduler (e.g.; LSF, SGE or gLite);
- network URL of the scientific service binary;
- endpoint (usually the BPEL process) to use for the asynchronous notifications from the scientific service;
- resource requirements; and
- authentication credentials (e.g., anonymous user, digitally signed security token, Grid proxy certificate).

Resource characteristic specification is, in fact, minimal and depends on the resource type (e.g., Grid rather than LSF); it should be more general and resource independent.

Fig. 5 shows an XML fragment describing a resource request.

```
<BSRequest>
  <BSScheduler>LSF</BSScheduler>
  <BSURL>http://www.dsf.unica.it/~andrea/PrimeNumber.jar</BSURL>
  <BSNotificationEndpoint>
    <add:EndpointReference
      xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
      <add:Address>
        http://10.12.0.6:8100/PrimeNumberNotification
      </add:Address>
      <wsa:ServiceName portName="PrimeNumberNotificationPort"
        xmlns:pn="http://wsdl.prime.cybersar">
        pn:PrimeNumberNotification
      </wsa:ServiceName>
    </add:EndpointReference>
  </BSNotificationEndpoint>
  <BSRequirements>...</BSRequirements>
  <BSCredentials>
    <NoAuth/>
  </BSCredentials>
</BSRequest>
```

Fig. 5. The XML fragment describing a resource request.

5.2. Resource access and provisioning

When a BPEL process wishes to execute a scientific business service, a resource is provisioned dynamically by the Resource Allocation service and an endpoint reference is generated accordingly. Resources typically live on private networks and can access the Internet through NAT servers, but cannot be accessed from the Internet. BPEL processes can run anywhere on the network, so a transparent proxy mechanism is required, both for the invocation of business scientific services (invoke BPEL activities) and for notifications from the same services (receive BPEL activities). Transparency is important, since BPEL processes should be unaware of such complications, and a simple and effective solution is to replace the real private endpoint reference of the scientific service with the public endpoint reference of the Proxy service, before notifying it to BPEL. The Proxy acts as an intermediary and routes the messages to the right destinations. In such a way the Proxy can monitor the flow of messages and can also detect network problems. Scalability is not a real issue because multiple Proxy services can be deployed simultaneously.

Infrastructure services are responsible for resource procurement towards BPEL processes. Global services invoke local services for specific types of resource, and the mission of the local services is to interface with lower-level middleware and/or APIs. Often, existing middleware and APIs are not WS aware, and proprietary interfaces must be used. In our current implementation,

- access to gLite middleware is performed by means of the opensource jLite [29] library, and
- access to LSF and SGE schedulers is done through the respective command line interfaces.

5.3. Authentication

Some of the security and authentication issues are discussed at the end of Section 4. Here we give the details related to our environment.

- gLite authentication: a proxy user certificate is generated through gLite or jLite command line interfaces and included in the resource request.
- LSF and SGE authentication: an LDAP server provides passwordbased user authentication, but this is not suitable over a network, and we have added public key certificates for our users to the LDAP server. User authentication is then performed by verifying the digital signature placed on an authentication token provided by the Resource Allocation service upon user request.

5.4. Data management

As we have already discussed, the pipelining of data-intensive scientific services can pose a serious limitation to BPEL orchestration, if data go back and forth inside SOAP messages. The Storage Manager global service provides storage resources for temporary data. Scientific services can read and write data directly over the network to the Storage Manager, without the need of sending them through BPEL processes. The Storage Manager is a service that implements simple but effective HTTP POST uploads and HTTP GET downloads. Uploads are available only during resource set lifetime. Multiple Storage Managers can coexist if scalability is an issue.

5.5. Execution monitoring

BPEL engines may provide facilities for monitoring service execution, but often they are difficult to use and are not effective. The general problem of monitoring, fault detection and compensation is still completely open, and here we are not proposing a solution, but a tool to aid users in identifying problems that prevent correct workflow completion. We have implemented a simple Monitor service which, based on the resource context information collected from the Resource Allocation and Proxy services, shows some facts about the execution of scientific business services, such as resource identifier, status, WSDL port type or last invocation/notification.

6. A collaborative environment for bioinformatics

To illustrate how a scientific experiment can be formalized and executed in the proposed scientific VO, let us consider a case study from the bioinformatics field. In particular, we focus on processing micro-array data, since they exemplify a situation that will be increasingly common in applications of machine learning to molecular biology.

6.1. Background

Micro-array technology enables to put the probes for the genes of an entire genome onto a chip, such that each data point provided by an experimenter lies in the high-dimensional space defined by the size of the genome under investigation. Because of the high cost of micro-array production, the number of replicas in these experiments is often severely limited, and many laboratories disseminate their results in the form of public datasets.

Related data are usually N x M matrices, where N is the number of genes (typically N > 10 000) and M is the number of samples (M < 100). As such, micro-array data are prime examples of extremely small sample-size but high-dimensional data. The large volume of data generated by experiments utilizing such essays, as well as the relatively high experimental noise often associated with them, require a careful computational analysis.

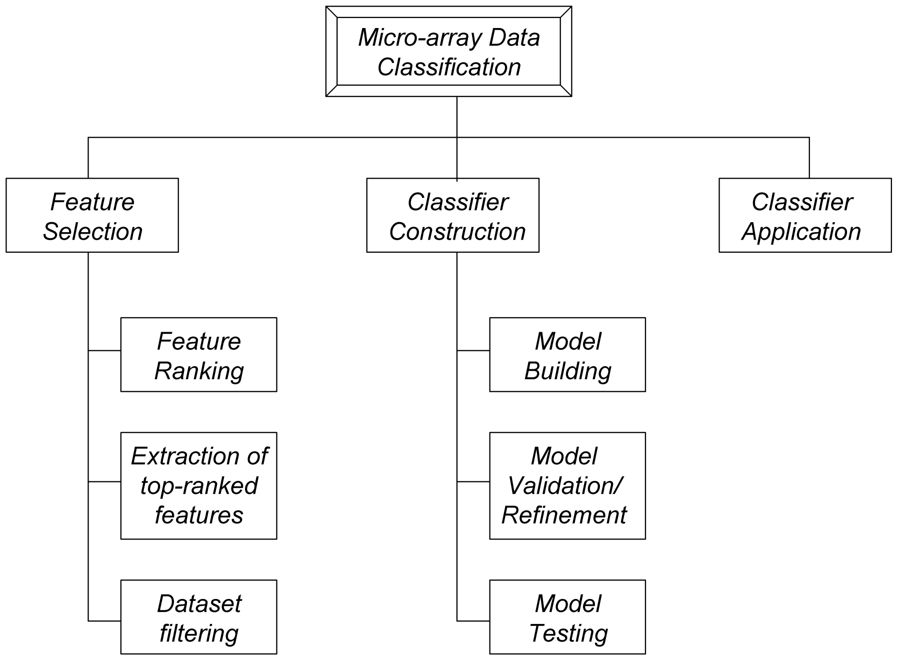

Fig. 6. The schema of the micro-array classification process.

Gene expression data generated using micro-arrays is generally used to classify the biologic samples based on the pattern of expression of all or a subset of genes on the micro-array. Although classification is by no means a new subject in the literature, the large and complex multivariate datasets generated by microarray experiments raise new methodological and computational challenges. Excluding a few special cases, finding good classifiers is known to be a very difficult task in the context of tumour classifications where gene expression profiles are used as complex biomarkers.

The construction of a classifier generally proceeds by selecting an informative set of genes (features or attributes) that can distinguish between the various classes (feature selection), choosing an appropriate mathematical model for the classifier and estimating parameters of the model based on the “training dataset” of tissues whose classification we know in advance (classifier training) [30,31]. Finally, the specificity and the sensitivity of the classifier are tested on the “test dataset”, i.e., a set of data that was not used in the process of constructing the classifier.

Fig. 6 shows the schema of the micro-array data classification experiment we consider. In more detail, feature selection can be decomposed into the following.

- Feature ranking, which orders features according to their predictive power (by applying some statistical or entropic evaluation measure).
- Extraction of top-ranked features, which implies defining a threshold (often based on domain expert experience) to cut off the ordered list of ranked features.
- Dataset filtering, which reduces the size of the original dataset by keeping only selected features and removing all the others. In addition, there are three separate tasks in classifier training.
- Model building, which builds a classification model for a given training dataset, using classification algorithms such as Bayesian Networks, Support Vector Machines, k-Near-

est- Neighbour, etc.

- Model validation/refinement, which adjusts the model parameters in order to obtain its best performance, as evaluated on a proper validation dataset (which can be extracted from the original training dataset).
- Model testing, which evaluates the model performance on an independent test dataset.

Finally, in Classifier Application, the classification model is applied to a set of unlabeled data, in order to make a prediction of the class (label) of each new instance.

6.2. Micro-array classification in SOA-enabled cooperative environments

The utilization of the SOA paradigm for the classification experiments described in the previous subsection includes Web Services acting both as data providers for public micro-array datasets and functionality providers for machine learning and other artificial intelligence techniques that are widely used and combined to classify gene expression data. At a minimum, the environment should provide the following categories of services.

- *Data extraction services*: a wide variety of useful scientific and in particular biological datasets are available on the web for classification experiments. Data can be downloaded from the repositories provided by many organizations and identified through an URL address, and usually differ in their format (i.e., RES, GCT, CLS, etc.). Data extraction services deal with data preprocessing in order to map the datasets of interest into a format suitable for the classification process.
- *Data mining services*: in terms of experiment design, the classification process can be partitioned into functional modules, which we call experiment services, where single computing functions are isolated and exposed as specific services to facilitate modularity and reuse. Typical experiment services are data mining algorithms and feature selection procedures both available as open-source or proprietary tools.
- *Visualization services*; these deal with the proper visualization of the experiment results in graphs and/or textual descriptions.

When running micro-array classification experiments, file staging mechanisms must be used for the processing of very large datasets. Moreover, methods for finding good classifiers are computationally intensive: comparing classifiers and selecting the best for each micro-array dataset can be a non-straightforward task. Consequently, the researcher is interested in combining different algorithms and reusing some pieces of previous work to extract useful knowledge from data. As such, the utilization of a workflow system in an SOA environment provides some advantages with regard to experiment orchestration and distributed computing.

6.3. Running micro-array classification experiments

The scientific business services developed in the context of the chosen case study, presented in detail in the previous subsections, are as follows.

- *Weka Data Mining*: a data mining service providing a subset of the functionalities exposed by the Weka library [32], such as feature ranking/selection, dataset filtering, classifier training, classifier testing.
- *Bar Chart*: a graphical visualization service, which shows the comparison between the accuracies of different classifiers in the form a bar chart graph.
- *Text Viewer*: a textual visualization service, which can be used to show confusion matrices and other measures related to classifier test results.

At this point, most of the pieces of our puzzle have been properly set in place; we are only missing a BPEL engine to orchestrate them. We have tested a couple of engines, namely Sun GlassFish v2 [33] in connection with NetBeans 6.5.1 [34], which provides a graphical BPEL editor (incredibly and unfortunately in the newer releases SOA support has been removed), and Apache ODE 2.0b2 [35] deployed into Apache Tomcat 6.0.26 [36].

The interaction diagram in Fig. 7 shows a flow of execution which illustrates the basic interaction between the SOA components, i.e., researcher, BPEL process, infrastructural services and scientific business services, and which is implemented in all BPEL processes (obviously in much more complex and interesting ways).

As a practical example, we consider a researcher interested in running a BPEL process which orchestrates scientific services in such a way to perform the following experiment: retrieve the training and test datasets of interest from a public repository (network URL), build and compare the accuracy of two different classifiers for an increasing number of selected attributes (by some attribute selection algorithm), and view the accuracies in a bar chart diagram as the number of attributes increases. The BPEL process may be already available or may be created/modified by the researcher and deployed to the preferred engine.

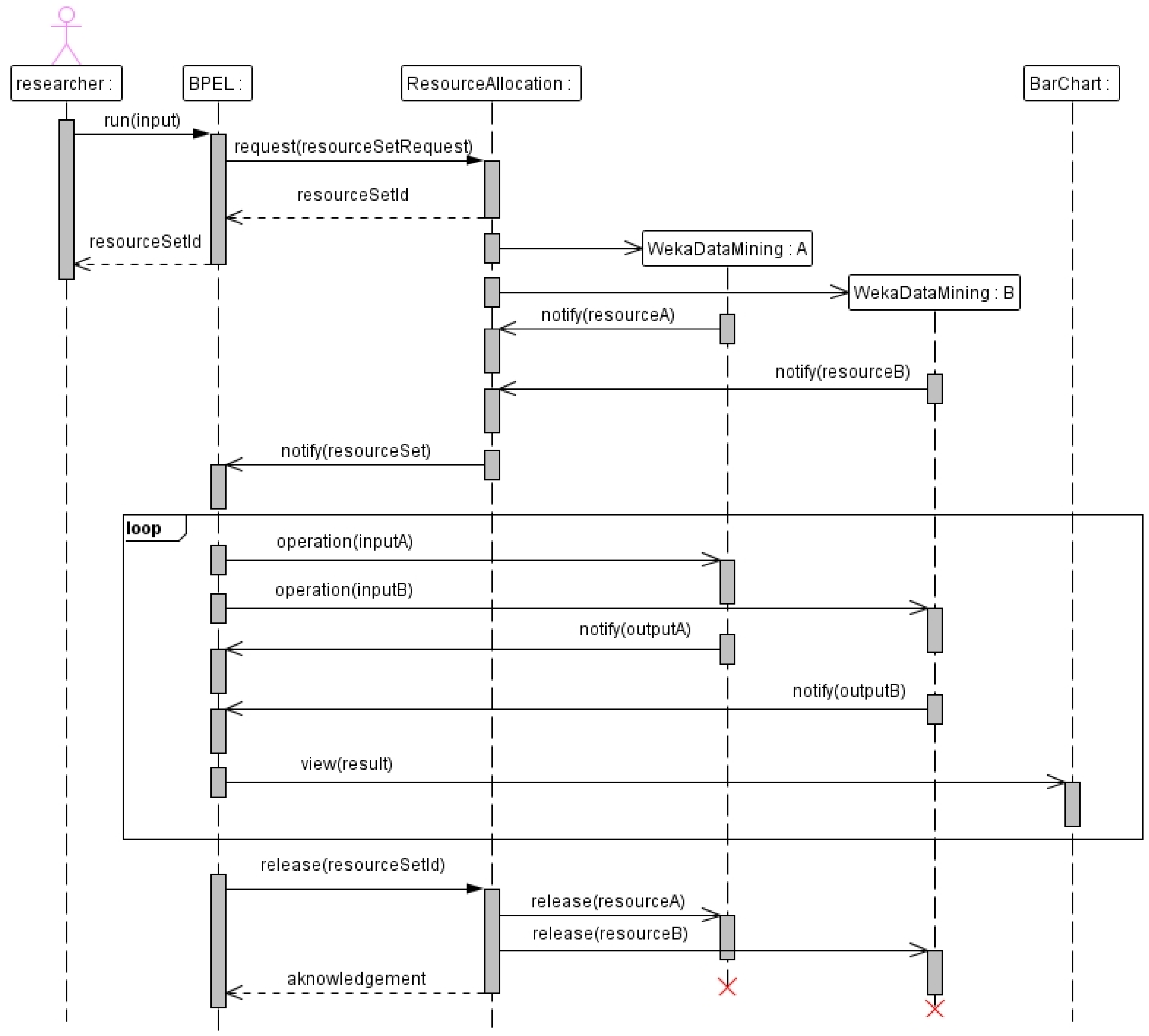

Fig. 7. Interaction between SOA components.

The resources required to perform the experiment are the following:

- a graphical desktop workstation for running the visualization service, which is started manually by the researcher;
- one computing resource for every instance of the data mining service, provisioned on demand by the Resource Allocation service; and
- storage resources for storing temporary results, provided by the Storage Manager service.

To illustrate the existence of different authentication policies, two categories of researchers or users are considered:

- Users exhibiting a valid gLite public key certificate issued by a VO federated with IGI and willing to use available IGI resources; such users are required to supply a valid proxy certificate.
- Anonymous users aiming at carrying out some tests on the resources of our local LSF cluster; anonymous users do not supply any credentials.

The input message to the BPEL process provides:

- Input data such as classifier names, URL of training and test dataset, etc.
- Resource set request (Section 5.1), containing the list of resources, URL of scientific service binaries, authentication credentials, etc.

After receiving the input file, the BPEL engine creates a new workflow instance performing the following activities.

1. Invoke (request-response) RA with resource set request:
   a) RA invokes RM responsible for gLite/LSF.
   b) RM invokes SM to validate authentication credentials; if authentication is successful the proper job description is formatted and submitted to the gLite/LSF for scheduling; this is done for every request in the set.
   c) RA returns the resource set identifier.
2. Receive (asynchronous) the resource set availability notification from RA:
   a) GLite/LSF schedules the jobs, and data mining service binaries are downloaded from the specified URL and started.
   b) RA is notified of data mining services start-up and of their context information (private endpoint references, etc.).
   c) RA sends to BPEL the list of resource identifiers and public endpoint references of data mining services.
3. Assign the public endpoint references to the corresponding partner links.
4. Invoke (request-response) bar chart visualization service with start-up data:
   a) Visualization service initializes a new chart and returns its identifier.
5. Loop over increasing number of features.
6. Invoke (one-way) data mining service operations:
   a) Proxy receives input messages and routes them to the correct data mining services.
   b) (Optionally) data mining services read input (e.g., datasets) from StM.
   c) Data mining services start the computation.
   d) Data mining services write output data to StM.
7. Receive (asynchronous) output messages from data mining services:
   a) Proxy receives output messages form the data mining services and routes them to BPEL.
8. Invoke (one-way) visualization service with chart identifier and computed classifier accuracy:

    a) Visualization service updates the specified chart with provided data.
9. End of loop.
10. Invoke (request-response) RA with resource set release request:
    a) RA releases the specified resource set and returns an acknowledgement.

Indeed, steps 6 and 7 are multiple: feature selection, dataset filtering, classifier training and classifier test operations are executed in sequential order; the same sequences of operations, executed for different classifiers, are instead executed in parallel (flow activities).

7. Related work

The term cooperative system is used to denote distributed information systems that are employed by Virtual Organizations. An extension of the cooperative paradigm, referred to as e-Applications, is becoming more and more frequent: e-Applications allow the dynamic composition of services provided by different organizations on the network. In addition to geographical distribution and interorganization cooperation, in e-Applications (i) cooperating organizations may not know each other in advance and (ii) services can be composed both at design and at runtime. E-Applications are a suitable mechanism for enabling coordination of heterogeneous actors in open environments [37]. Taking into account many concepts of Human Organization Theory, models for cooperation have been developed [38,39] to describe the structural, functional, normative and environmental aspects of a system based on the main concepts of organizational units, services, norm and environment.

Among the technologies available for the development of business e-Applications, SOA [15] and in particular Web Services [16] offer a promising approach [40,41] supporting enterprise VOs, and are among those which can support the development of cooperative information systems, as extensively discussed in [42]. The integration of such technologies within a research environment requires a clear understanding of how scientific collaborations are performed through activities whose underlying organizational dynamics is similar to the processes occurring in distributed cooperative enterprise units. Key aspects of scientific workflows are presented in [18].

Software applications have been built to address a wide spectrum of scientific workflows, ranging from basic tools that are designed to handle "desktop" tasks such as simple data analysis and visualization to complex workflow systems that are designed to run large-scale e-Science applications on remote Grid resources. These systems need to support multiple concurrent users, deal with security requirements, and run workflows that may require the use of a sophisticated layer of services [20].

McPhillips et al. [18] identify desiderata for scientific workflow systems - namely clarity, predictability, reportability and reusability. Moreover, ease of composition and editing, the ability to automatically log and record workflow enactments and the flexibility to incorporate new tools are all important features [20]. The interoperability aspects of scientific workflow systems are addressed in [17], which investigates the differences in execution environments for local workflows and those executing on remote Grid resources. A complete overview of features and capabilities of scientific workflow systems is presented in [3].

There are a number of widely recognized Grid workflow projects. Many of these began life in the "desktop" workflow space, but they have evolved over time to address the large-scale e-Science applications. A Grid-aware framework for the construction of distributed workflows and their management and execution is provided by systems like Triana [43], Kepler [9], Pegasus [5], and ASKALON [7]. Specifically designed for the life sciences, Taverna [11] was the first system to recognize the importance of data provenance and semantic Grid issues. Based on BPEL [44], QoWL [45] and GPEL [46] are significant examples of workflow systems designed for dynamic, adaptive large-scale e-Science applications.

In particular, BPEL is recognized [3] as the de facto standard for Web-Service-based workflows. The use of BPEL for Grid service orchestration is proposed as foundation in [47] since it fulfils many requirements of the WSRF standard. The appropriateness of BPEL is also examined and confirmed in [48-50]. These works mainly focus on scientific workflows and rely on extending or adapting BPEL, thus creating dialects. While developed for the business domain, technologies like BPEL are then recognized suitable to address the requirements of e-Science applications [19], supporting the composition of large computational and data analysis tasks that must execute on remote supercomputing resources.

An architecture for the dynamic scheduling of workflow service calls is presented in [51], where control mechanisms of BPEL are combined with an adaptive runtime environment that integrates dedicated resources and on-demand resources provided by infrastructures like Amazon Elastic Compute Cloud. Experience with adapting a WS-BPEL runtime for Grid workflows is presented in [52,53]. [54] presents BPEL extensions to invoke stateful Web Services that are widely used in a Grid middleware.

Based on BPEL, a Grid-enabled workflow managed system is presented in [55] in which the service binding at runtime is achieved in a way similar to ours: a provisioning service looks up the resource where the requested service is installed. However, it does not provide mechanisms for dynamically procurement of additional target resources.

The lack of a standard generic architecture for resource provisioning is discussed in [56]. Similar to our infrastructural services, the authors present a set of services for resource procurement and BPEL-based workflows that make use of these services.

8. Conclusions

E-Science applications require the ability to perform long-lived, peer-to-peer collaborations between the participating researchers. In this paper, we have demonstrated that combining Web Services, Grids and other distributed resources is a promising way to leverage on existing work in both business and scientific environments, since Web Service specifications offer a communication bridge between the heterogeneous computational environments used to develop and host applications.

Starting from the analysis of the basic issues of the SOA paradigm in addressing the needs of e-Science environments, we presented the main features of an SOA-enabled scientific VO. Since we expect that workflows will have a role as important as they already have in business environment, we place particular emphasis in their capabilities of managing and integrating distributed resources and scientific applications.

The main advantages of this kind of approach are as follows.

- Efficiency: the procedures with high support (help to discover and localize the resources) free the workflow designer from technical and repetitive work and this contributes to the creation of "best practices" that are eventually valuable, comparable and can be shared with other people.
- Reproducibility: the workflow can be repeated with both automation and precision in the time, and also by a third party, on instances of different data and parameters.
- Traceability: the workflow is followed in an environment where the data source can be traced and checked a priori.

The proposed approach is quite general and flexible. It has been applied to bioinformatics environments and has returned a set of results and feedbacks regarding its implementation, usage, and the overall feasibility. However, application scenarios are not limited and span brain dynamics investigation, geo-informatics and drug discovery, as well as improving our current work on micro-array data analysis.

The choice of complementing standard BPEL with a set of infrastructural services, rather than developing new BPEL extensions, somehow limits the full potential offered by a direct access to WSRF-enabled Grid Services. Our goal was to suggest a direction towards open standards service-based systems for e-Science environments, and we believe that the compatibility and interoperability offered by BPEL will support the migration towards enhanced workflow systems built on concepts and technologies that are inherited both from Grid and Web Services communities.

Issues to be addressed, beyond the core issues of setting in place the proposed collaboration environment, arise in various areas, such as workflow execution monitoring, fault handling and compensation, scalability, policy enforcement, trust and security support, collaboration correctness monitoring, quality of service monitoring, transaction logging, and so on.

9. References

[1] D. De Roure, Y. Gil, J.A. Hendler (Eds.), e-Science, IEEE Intelligent Systems 19 (1) (2004) (special issue).

[2] I. Taylor, E. Deelman, D. Gannon, M. Shields (Eds.), Workflows for e-Science, Springer, New York, Secaucus, NJ, USA, 2007.

[3] E. Deelman, D. Gannon, M. Shields, I. Taylor, Workflows and e-Science: an overview of workflow system features, Future Generation Computer Systems 25 (2009) 528-540.

[4] E. Elmroth, F. Hernandez, J. Tordsson, A light-weight Grid workflow execution engine enabling client and middleware independence, in: R. Wyrzykowski, et al. (Eds.), Parallel Processing and Applied Mathematics, 7th Int. Conference, PPAM 2007, in: LNCS, vol. 4967, Springer Verlag, 2008, pp. 29-270.

[5] E. Deelman, et al., Pegasus: a framework for mapping complex scientific workflows onto distributed systems, Science Program 13 (3) (2005) 219-237.

[6] P. Kacsuk, et al., P-GRADE: a Grid programming environment, Journal of Grid Computing 1 (2) (2003) 171-197.

[7] T. Fahringer, et al., ASKALON: a development and Grid computing environment for scientific workflows, in: I. Taylor, E. Deelman, D. Gannon, M. Shields (Eds.), Workflows for eScience: Scientific Workflow for Grids, Springer-Verlag, 2007.

[8] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, I. Wang, Programming scientific and distributed workflows with Triana services, Concurrency and Computation: Practice and Experience 18 (10) (2006) 1021-1037.

[9] D.D. Pennington, et al., Ecological Niche modeling using the Kepler workflow system, in: I. Taylor, E. Deelman, D. Gannon, M. Shields (Eds.), Workflows for eScience: Scientific Workflow for Grids, Springer-Verlag, 2007.

[10] T. Oinn, et al., Taverna: a tool for the composition and enactment of bioinformatics workflows, Bioinformatics 20 (17) (2004) 3045-3054.

[11] T. Oinn, et al., Taverna/myGrid: aligning a workflow system with the life sciences community, in: I. Taylor, E. Deelman, D. Gannon, M. Shields (Eds.), Workflows for eScience: Scientific Workflow for Grids, Springer-Verlag, 2007.

[12] D. De Roure, C. Goble, R. Stevens, The design and the realization of the my experiment virtual research environment for social sharing of workflows, Future Generation Computer Systems 25 (2009) 561-567.

[13] G. Folino, A. Forestiero, G. Papuzzo, G. Spezzano, A Grid portal for solving geoscience problems using distributed knowledge discovery services, Future Generation Computer Systems 26 (2010) 87-96.

[14] M.P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, Service-oriented computing: state of the art and research challenges, IEEE Computer (2007).

[15] M.P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, Service-oriented computing: a research roadmap, International Journal of Cooperative Information Systems 17 (2008) 223-255.

[16] G. Alonso, F. Casati, H.A. Kuno, V. Machiraju, Web Services - Concepts, Architectures and Applications, Springer, 2004.

[17] E. Elmroth, F. Hernandez, J. Tordsson, Three fundamental dimensions of scientific workflow interoperability: model of computation, language and execution environment, Future Generation Computer Systems 26 (2010) 245-256.

[18] T. McPhillips, S. Bowers, D. Zinn, B. Ludascher, Scientific workflows for mere mortals, Future Generation Computer Systems 25 (2009) 541-551.

[19] A. Akram, D. Meredith, R. Allan, Evaluation of BPEL to scientific workflows, in: Proceedings of CCGRID'06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 269-274.

[20] G. Fox, D. Gannon, A survey of the role and use of web services and service oriented architectures in scientific/technical Grids, Technical Report, Indiana University, 2006.

[21] http://www.platform.com/workload-management/high-performancecomputing.

[22] http://www.sun.com/software/sge/.

[23] http://glite.web.cern.ch/glite/.

[24] http://www.globus.org/.

[25] http://www.ogf.org/documents/GFD.72.pdf.

[26] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf.

[27] http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html.

[28] http://www.italiangrid.org/.

[29] http://code.google.com/p/jlite.

[30] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Machine Learning 46 (1-3) (2002) 389-422.

[31] N. Dessì, B. Pes, Framework for multi-class learning in micro-array data analysis, in: Lecture Notes in Computer Science, vol. 5651, 2009, pp. 275-284.

[32] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten, The WEKA data mining software: an update, SIGKDD Explorations 11 (1) (2009).

[33] https://glassfish.dev.java.net/.

[34] http://netbeans.org/.

[35] http://ode.apache.org/.

[36] http://tomcat.apache.org/.

[37] Americo Nobre G.G. Amorim, Virtual organization theory: current status and demands, in: Integration and Innovation Orient to E-Society, Volume 1, vol. 251, Springer, Boston, 2008, pp. 1-8.

[38] J.T. Pollock, R. Hodgson, Adaptive information: improving business through semantic interoperability, in: Grid Computing, and Enterprise Integration, in: Wiley Series in Systems Engineering and Management, Wiley-Interscience, 2004.

[39] N. Criado, E. Argente, V. Julián, V. Botti, Designing virtual organizations in: Advances in Soft Computing, Proc. of 7th International Conference on Practical Applications of Agents and Multi-Agent Systems, PAAMS 2009, Springer Verlag, 2009, pp. 440-449.

[40] A. Bosin, N. Dessì, M. Fugini, B. Pes, Cooperative e-organizations for distributed bioinformatics experiments, in: Lecture Notes in Computer Science, vol. 5326, 2008, pp. 306-313.

[41] A. Bosin, N. Dessì, B. Madusudhanan, B. Pes, Will SOA accommodate the next step of e-Science?, in: Notere, 10th Annual International Conference on New Technologies of Distributed Systems, Tozeur, Tunisia, May 2010, pp. 51-58.

[42] E. Di Nitto, J. Mylopoulos, M. Papazoglou (Eds.), At Your Service: An Overview of Results of Projects in the Field of Service Engineering of the IST Program, in: Series on Information Systems, MIT Press, 2009.

[43] I. Taylor, M. Shields, I. Wang, A. Harrison, Visual Grid workflow in Triana, Journal of Grid Computing 3 (3-4) (2005) 153-169.

[44] http://www.oasis-open.org/committees/wsbpel.

[45] I. Brandic, S. Pllana, S. Benkner, High-level composition of QoS-aware Grid workflows: an approach that considers location affinity, in: Proceedings of WORKS06, Paris, 2006.

[46] Slominski, Adapting BPEL to scientific workflows, in: I. Taylar, E. Deelman, D. Gannon, M. Shields (Eds.), Workflows for eScience: Scientific Workflow for Grids, Springer-Verlag, 2007.

[47] F. Leymann, Choreography for the Grid: towards fitting BPEL to the resource framework: research articles, Concurrency and Computation: Practice and Experience 18 (10) (2006) 1201-1217.

[48] K. Chao, et al., Analysis of Grid service composition with BPEL4WS, in: Proceedings of AINA'04, vol. 1, IEEE Computer Society, Los Alamitos, CA, USA, 2004, p. 284.

[49] T. Dornemann, et al., Grid workflow modelling using Grid-specific BPEL extensions, in: German e-Science, 2007.

[50] W. Emmerich, et al., Grid service orchestration using the business process execution language, BPEL, Journal of Grid Computing (2006) 283-304.

[51] T. Dornemann, E. Juhnke, B. Freisleben, On-demand resource provisioning for BPEL workflows using Amazon's elastic compute cloud, in: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE, 2009, pp. 140-147.

[52] Di Penta, et al., WS binder: a framework to enable dynamic binding of composite web services, in: International Workshop on Service Oriented Software Engineering, ACM, 2006, pp. 74-80.

[53] A. Bosin, N. Dessì, B. Madusudhanan, A service-based approach for the execution of scientific workflows in Grids, in: Conference on Computing Frontiers, 2010, pp. 107-108.

[54] T. Dornemann, T. Friese, S. Herdt, E. Juhnke, B. Freisleben, Grid workflow modelling using Grid specific BPEL extensions, in: Proceedings of German e- Science Conference, GES, 2007, pp. 1-8.

[55] R.Y. Ma, Y.W. Wu, X.X. Meng, S.J. Liu, L. Pan, Grid-enabled workflow management system based on BPEL, International Journal of High Performance Computing Applications 22 (3) (2008) 238-249.

[56] R. Mietzner, F. Leymann, Towards provisioning the cloud:on the usage of multigranularity flows and services to realize a unified provisioning infrastructure for SaaS applications, in: Proceedings of IEEE Congress on Services, Los Alamitos, CA, USA, 2008, pp. 3-10.