On-Demand Service Deployment and Process Support in e-Science DLs: the Diligent Experience*

Leonardo Candela[1], Donatella Castelli[1], Christoph Langguth[2], Pasquale Pagano[1], Heiko Schuldt[2], Manuele Simi[1], and Laura Voicu[2]

[1] Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo" – CNR
Via G. Moruzzi, 1 – 56124 Pisa – Italy
{candela|castelli|pagano|simi}@isti.cnr.it
[2] University of Basel – Department of Computer Science
Bernoullistrasse 16 – 4056 Basel – Switzerland
{christoph.langguth|heiko.schuldt|laura.voicu}@unibas.ch

Abstract. Digital Libraries (DLs) are nowadays conceived as integrated tools promoting knowledge creation and consumption. They are moving far from the connotation of the term "library" to become a co-operative environment supporting all the actors involved in the whole process of knowledge creation, management, exchange, etc. Due to the different application contexts in which these systems have to be applied and their demanding requirements, novel and flexible strategies for developing, creating, and maintaining DLs are needed. This paper focuses on the DILIGENT approach where Digital Libraries are created on-demand by promoting resource sharing and where complex DL applications can be supported by the on-demand, application-specific combination of basic services. The paper discusses the technical solution by presenting the system architecture providing details about its key components and reports on its ongoing exploitation to fulfil the application scenario arising in the earth observation application context.

1 Introduction

Nowadays research is a cooperative and multidisciplinary effort carried out by groups belonging to different organisations distributed over the world. Essentially, cooperation is strongly based on digital counterparts of classical documents, like PDF files, complex (multimedia) objects combining text, images, audio and/or video, sensor data, structured and semi-structured data, etc., residing in various kinds of information sources. Research groups thus have to dynamically aggregate into *virtual research organisations* with the aim to share their digital resources for the duration of their collaboration creating thus new and powerful research environments. In this collaborative scenario, virtual research organizations increasingly require Digital Libraries (DLs) as tools supporting and accelerating the achievements of their research results. The DLs required by these organizations are far beyond any connotation of the traditional term "library". In particular, these new tools must be able to: (i) manage any type of information deemed as relevant with respect to the research;

* This work is partially funded by the European Commission in the context of the DILIGENT project (IST-2003-0042690), under the 2^{nd} call of the FP6 IST priority.

(*ii*) support the entire information life cycle, from the creation to the exploration, manipulation and exchange in a collaborative manner; (*iii*) provide mechanisms for the controlled and personalised sharing of the research products under policies established by their producers; and finally, (*iv*) evolve on-demand to adapt themselves to the changes of both the external environment and the community internal needs.

Resource sharing in eScience Digital Libraries is strongly facilitated by the recent advances in *Grid computing* [6,7,8,9]. The nature of the resources shared can be highly heterogeneous. They range from computing and storage capacity (computational and storage Grid, respectively) to services providing enhanced functionality, like access to expensive instruments and archives containing scientific data, reports, and any sort of knowledge deemed as relevant with respect to their research (service Grid).

In the new framework established by these DLs, a community of researchers working on a same project can decide to set up a DL for supporting their temporary scientific collaboration by exploiting selected resources to the community. The researchers can use these DLs both as a source of information for supporting their research work and a means for publishing and disseminating their research outcomes to colleagues. By exploiting a DL a researcher, for example, can retrieve information pertinent to her experimental study and communicate the results of her research by publishing in the library a composite and living document that contains: (*i*) the textual formulation of the thesis statement; (*ii*) a description of the conditions under which the experiments have been carried out, given in terms of images and videos, selected from different DL repositories, illustrating the tools used and the experimental environment; and (*iii*) a "live graph", i.e., a graph which is updated each time the information object is accessed showing the results of her experiments.

Services for the creation, manipulation, sharing, management, etc. of digital objects are crucial to eScience DLs. In addition to managing these services (some of them are computationally intensive) and to deploying them on demand in a Grid, their combination into *processes* or *workflows*, e.g., to elaborate data extracted from several large raw data archives, allows building complex DL applications. Consider as a sample application the generation of complex reports for earth observation. After accessing several raw data archives (by means of appropriate services), data first needs to be transformed into a common format and possibly also aggregated. Next, the combination of data from different sources has to be accomplished (e.g., to create an overlay image of the observed area). Finally, this derived and combined data needs to be incorporated into a report document. Each time this report is accessed, the embedded data is updated by invoking this sequence of services.

The realization of DLs and process-based DL applications serving the new e-Science scenario illustrated by the example above requires an in depth rethinking of DL technology. In this paper we focus our attention on the mechanisms that a DL system must provide in order to support the on-demand deployment of services and the support for complex workflows in a Grid-based e-Science DL environment. In terms of processes, their definition, validation and execution needs to be supported, thereby implementing new (compound) services. In order to avoid bottlenecks and to guarantee a high degree of scalability, DL applications needs the possibility to deploy new services on demand. This, in turn, requires the continuous observation of the state of existing services. Thus, the functionality of a DL is obtained by appropriately aggregating and combining available services. Such DLs must be capable

to easily adapt to and fulfil the evolving requirements of their community, e.g., a newly available information source should be easily added to the DL if it is deemed relevant, additional instances of a certain service may be deployed on new hosting nodes to respond to a temporary increased workload, or a new service may be included to provide a new functionality. This paper focuses on the DILIGENT [3] approach to Grid-enabled Digital Libraries. DILIGENT is an ongoing EU IST project aiming at providing a test-bed enabling members of e-Science organisations to dynamically create the DLs they need by accessing shared knowledge and available tools and services. This test-bed is implemented by integrating digital libraries with Grid technologies.

This paper is organised as follows: Section 2 presents the DILIGENT infrastructure by providing an overview of its system architecture. Section 3 concentrates on two key services supporting the operation of the DILIGENT service-oriented infrastructure. Section 4 provides details about the pool of services dedicated to services composition and thus to the mechanisms supporting the deliver of novel functionality. Section 5 presents an exploitation of the DILIGENT infrastructure in fulfilling the concrete scenario arising in the earth observation community. Finally, Section 6 concludes.

2 DILIGENT: An Infrastructure Supporting e-Science DLs

2.1 DILIGENT Overview

DILIGENT is an ongoing IST project that aims to combine Grid [6,7,8,9] and Digital Library [1,10,11,14] technologies in order to provide an advanced test-bed DL infrastructure allowing members of dynamic virtual e-Science organizations to access shared knowledge and to collaborate in a secure, coordinated, dynamic and cost-effective way. In particular, DILIGENT builds on top of the Enabling Grid for E-sciencE (EGEE) [4] framework which is developing the largest Grid Infrastructure and which, in turn, exploits the EU Research Network GÉANT [12].

From an abstract point of view, the DILIGENT infrastructure acts as a *DL broker*, where the clients of the broker are DL resource providers and consumers. The providers are the individuals and the organisations that decide to make available, under the supervision of the infrastructure, their resources according to certain access and use policies. The consumers are the user communities that want to build their own DLs. The resources managed by this broker are of different types: *collections* (i.e., set of information objects searchable and accessible through a single "access point"), *services* (i.e., software tools implementing a specific functionality and whose descriptions, interfaces and bindings are defined and publicly available), *hosting nodes* (i.e., networked entities that offer computing and storage capabilities and supply an environment for hosting collections and services), and *EGEE resources* (i.e., computing elements and storage elements).

In order to support the controlled sharing of resources among providers and consumers, the DILIGENT infrastructure relies on the *virtual organizations* (VOs) mechanism that has been introduced in the Grid research area [9]. This mechanism models sets of users and resources aggregated together by highly controlled sharing rules, usually based on an authentication framework. VOs have a limited lifetime and are dynamically created to satisfy specific needs by allocating and providing resources on-demand.

By exploiting appropriate mechanisms provided by the infrastructure, providers register their resources by supplying a description of them. According to the type of resources provided, the infrastructure also automatically extracts other properties that are used to enrich the explicit description. The infrastructure takes care of the management of the registered resources by supporting their discovery, monitoring, reservation, and by implementing the functionality needed to support the required controlled sharing and quality of service (see Section 3).

A user community can create one or more DLs by specifying a set of requirements and by appropriately combining the available resources. These requirements specify conditions on the information space (e.g., the set of collections, subject of the content, documents type), on the services for supporting the work of the users (e.g., type of search), on the quality of service (e.g., availability, performance, security) and on many other aspects, like the maximum cost, lifetime, etc. The DL broker satisfies the given requirements by selecting, and in many cases also deploying, a number of resources among those accessible to the community, gluing them appropriately and, finally, making the new DL application accessible through a portal. The composition of a DL is dynamic since the infrastructure continuously monitors the status of the DL resources and, if necessary, changes them in order to offer the best quality of service. Therefore, DLs (possibly serving different communities) can be created and modified *on-the-fly*, without considerable investments and changes in the organisations that set them up.

2.2 The DILIGENT Architecture

The DILIGENT infrastructure (depicted in Figure 1) is currently being constructed by implementing a service-oriented architecture in a Grid framework. The ovals in the figure represent functional areas while the boxes model the services of each area. These services rely on an *application framework* that is mainly constituted by (*i*) the *gLite* Grid middleware [5] which supports access to the EGEE resources, and (*ii*) the WSRF specification [2] implementation released by the Globus project [13]. By relying on such software framework, the DILIGENT (DL and non-DL) services are entitled to act as Grid Services and thus have access to shared resources in the Grid. In this framework, resources are either the computing elements and storage elements provided by the EGEE project or the Grid Services provided by DILIGENT.

The *Mediation* area includes a number of wrapper services. They are in charge of accessing external information sources in order to transform the external objects into DILIGENT information objects organised in collections.

The *Information Space Management* area first contains the *Content Management* services that encompass *Replication Management* and *Storage Management*. They represent the DILIGENT information objects repository. Second, the *Metadata Management* services that exploit the capabilities provided by the Content Management for the storage and management of the metadata manifestations. Third, the *Annotation service* which provides the functionality for the management of annotations attached to information objects. Finally, the *Content Security* service that takes care of applying watermark and encryption techniques in order to protect the DL information objects from unauthorised accesses. The latter aspect is particularly relevant since the Storage Management relies on the Grid storage facilities and thus

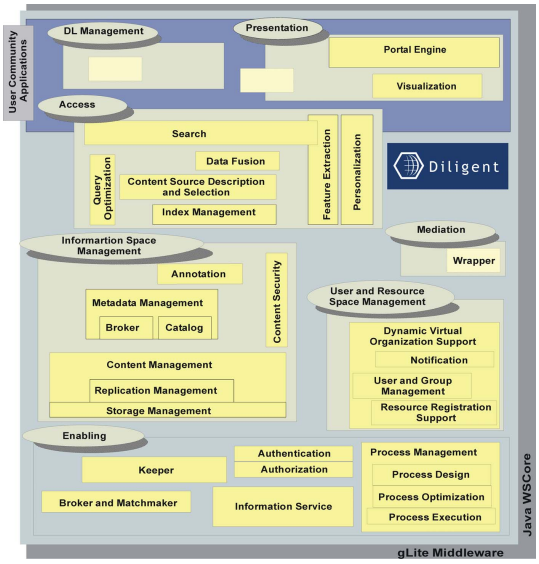

Fig. 1. The Logical Architecture of DILIGENT

stores physical files on third party devices that are not under the direct control of the DILIGENT infrastructure.

The *Access* area includes the *Search* service that exploits the capabilities provided by (*i*) *Index Management* which encompasses the component in charge of building and maintaining indexes of various types on the information domain, e.g., full-text and forward indexes; (*ii*) *Feature Extraction* that collects a number of features extraction components specialized for different media types; (*iii*) *Content Source Description and Selection*, i.e., the component supporting the discovery of the collections to be searched; (*iv*) *Data Fusion*, i.e., the component in charge of merging the result sets coming from different collections; and (*v*) *Personalization* which is a service in charge of customising the search results according to the user's preferences. All these services are designed to co-operate by relying on the Process Management facilities (more details are given in Section 4). In short, the timely reply to a query requires that a search service produces an execution plan involving all the needed services modeled by means of a workflow which will be executed in a Peer-to-Peer fashion by the Process Execution service.

The *User and Resource Space Management* area includes the *Dynamic Virtual Organization Support* service. Actually this is the "meta-service" in charge of supporting the implementation of the Virtual Organization which creates the trusted environment needed for ensuring a controlled sharing of the DILIGENT resources. In order to perform its task, the service provides and relies on a series of services: the *Notification* service which informs users about certain topics, e.g., availability of a new resource; the *User and Group Management* service which supports user registration and management tasks and provides mechanisms for their organization in groups; and *Resource Registration Support* services which allows adding new resources to the DILIGENT infrastructure.

The *Presentation* area is strongly user-oriented. It supports the automatic generation of user-community specific portals, thus providing personalised access to the DLs. In particular, it has been designed to support the plug and play of user communities specific visualization tools. From a technological point of view, DILIGENT uses an open-source portlet-hosting engine (GridSphere) that by relying on JSR168[3], JavaServer Faces[4] and WSRP[5] standards is capable of hosting the user interfaces that are integral parts of the DILIGENT services.

Finally, the services of the *Enabling* framework are in charge of providing functionality for (*i*) monitoring and discovering of all the available DILIGENT resources – *Information Service*; (*ii*) implementing a global strategy offering the optimal use of the hosting node resources supplied by the DILIGENT infrastructure – *Broker & Matchmaker Service*; (*iii*) orchestrating the pool of resources that populate the various virtual DLs and ensuring certain levels of fault tolerance and QoS – *Keeper Service*; (*iv*) supporting the design and verification of workflows, as well as services ensuring their reliable execution and optimization – *Process Management Service*.

In the remainder of this paper, we focus on the technical details of the latter aspects, namely support for on-demand deployment of services, automatic adaptation of the DL to changing environments and/or load characteristics, and complex, process-based applications.

3 Service Management

In a distributed infrastructure where the applications are provided by aggregation and composition of services, a set of facilities dedicated to their discovery and, in general, to their management are needed. In DILIGENT where the presence of the Grid promotes a dynamic development process – thus the dynamic deployment of novel resources – this need is even more urgent. In this section we introduce the DILIGENT services dedicated to provide such support.

3.1 The Information Service

The *Information Service (IS)* is the service in charge of supporting the discovery and continuous monitoring of distributed resources forming the infrastructure with the appropriate level of freshness. Hence, the infrastructure is able to adapt the usage of resources in a flexible way by dynamically balancing their load. The IS gathers and supplies information following an approach inspired by the well-known Grid Monitoring Architecture (GMA) [18] proposed by GGF[6] that models an information infrastructure as composed by a set of *producers* (that provide information), *consumers* (that request for information) and *registries* or collectors (that mediate the communication between producers and consumers).

In this scenario, depicted in Figure 2, producers and consumers are supported in interacting with the IS via a lightweight component distributed on each hosting node of the infrastructure, called *IS-Client*. This component provides three main kinds of

[3] `www.jcp.org/en/jsr/detail?id=168`
[4] `java.sun.com/javaee/javaserverfaces/`
[5] `www.oasis-open.org/committees/download.php/10539/wsrp-primer-1.0.html`
[6] Global Grid Forum, `www.ggf.org`

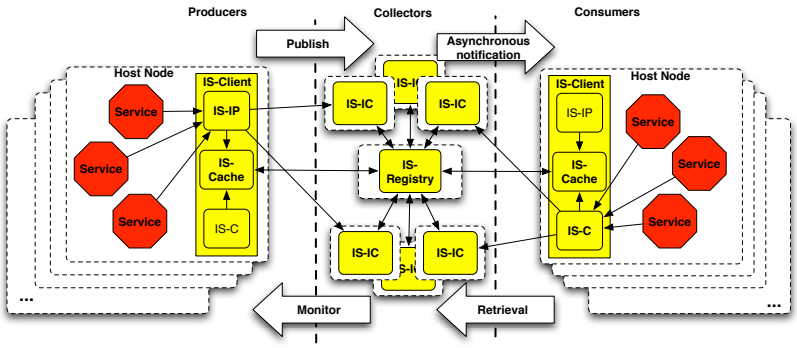

Fig. 2. The Logical Architecture of the Information Service

functionality. First, publication of the information (*IS-IP library*). Second, information access and discovery via query and subscription/notification mechanisms (*IS-C*). Third, local storage and maintenance of useful and constantly updated information (*IS-Cache*). The IS-Client implements an efficient access to and the publication of the information in the distributed infrastructure while hiding any detail of the routing process that identifies the appropriate collectors where such information is to be published in/retrieved from. The collector aggregates the produced information. It is composed of two components, namely the *IS-Registry* and the *IS-IC*. The former component acts as a classical registry and it is in charge of maintaining the list of available services and their static information while the latter maintains all the dynamic information and it is based on a highly distributed architecture. From an operational point of view, it is important to note that each time a service of the federation is deployed, it is registered on the IS-Registry and then it starts producing its dynamic information via the local IS-IP. In parallel, the IS-Cache takes care of maintaining the set of minimal information needed to the locally hosted services both for publishing and for querying. The IS-Registry is in charge of maintaining the "picture" of the whole infrastructure in line with the actual status by continuously monitoring the service instances.

From a technical point of view this services rely on the WS-* standards and specifications, namely the WSRF framework, WS-Addressing, WS-Security, and WS-Notification.

3.2 The Keeper Service

In the DILIGENT context where a DL is built by appropriately aggregating resources, the *Keeper* is the real manager. It is the service in charge of creating resources. In addition, via the creation of an appropriate virtual organisation, the Keeper is able to properly authorize the users of these services. In order to provide this facility, the Keeper relies on the concepts of *hosting node* and *software package*. The former is a DILIGENT resource capable of hosting services. The latter is a bunch of software that, once deployed by the Keeper, provides the functionality of a service.

The logical architecture of the Keeper is depicted in Fig. 3. It is composed of the *DL Management*, the *Hosting Node Manager* (HNM), and the *Packages Repository*.

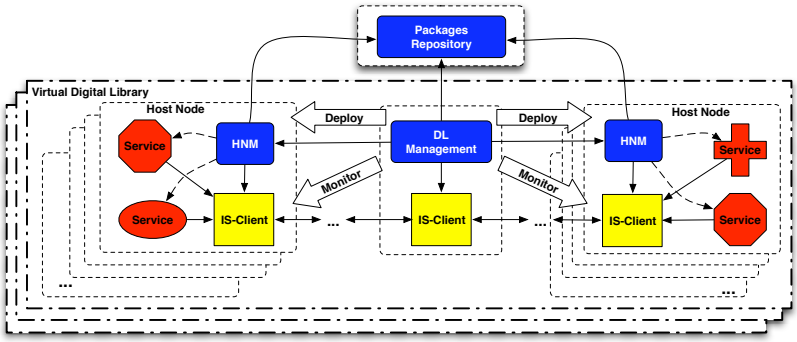

Fig. 3. The Keeper Service Logical Architecture

The task of DL Management is to identify the set of software packages needed to implement the DL and the set of hosting nodes where these software components will be deployed. By interacting with the HNM of these nodes, the DL Management directs the deployment. In addition, it coordinates and disseminates the *operational context* that transforms this set of distributed resources into a single application. In the DILIGENT terminology, this context is named *DL Map* and specifies the DL resource locations and their configurations. Any other dynamic information about a resource (e.g., its status) is maintained and disseminated by the IS.

Once the DL is up and running, the Keeper is also in charge of guaranteeing the quality of the overall set of DL functionality at any time by dynamically reallocating resources/archives and checking periodically their status. In order to support this functionality it accesses and investigates the state of services and resources and destroys and/or relocates them in an appropriate way using the information disseminated by the Information Service.

The HNM is the minimal mandatory software that must be installed on a hosting node to support the dynamic deployment. It is the manager of the node which hosts it. The tasks of node management are fourfold. First, it collaborates with the DL Management to deploy new services. Second, it publishes the hosting node configuration and status in the Information Service. Third, it exchanges data with the DL Management of the DL and fourth, it maintains and exposes the hosting node configuration to the hosted services.

Finally, the Packages Repository is in charge of storing the software packages and making them accessible to the HNMs when they need to deploy one of them.

From a technical point of view the resources we are discussing here are Web/Grid Service instances and related software components. All the software that we want to dynamically instantiate must be registered in the DILIGENT infrastructure and must be compliant with the *package model specification*. If a "piece of software" respects the rules of this specification, it can be (*i*) uploaded in the Packages Repository, (*ii*) handled – i.e., deployed and undeployed – by the HNM, and (*iii*) dynamically discovered and used by other services. Moreover, since DILIGENT follows the SOA paradigm, it is clearly composed of services that must be accessible via a network interface. A *hosting environment* is needed in order to allow services to operate. In

DILIGENT the hosting environment is the Java WS Core, developed by the Globus Project. This service container provides a complete implementation of the WSRF [2] and WS-Notification specifications plus WS-Addressing and WS-Security support based on the Axis Web Services engine developed by the Apache Foundation. Java WS Core provides also a framework that supports both authentication and authorization mechanisms. In particular, the authorisation features can be extended by each service in order to provide a customized level of authorisation policies.

4 Service Composition

Service composition allows for the (recursive) definition of complex services out of existing ones. This is often also referred to as *programming in the large*. For an eScience DL, service composition is particularly important in order to specify and run processes for the definition, analysis, and processing of data in several subsequent, individual steps.

A sophisticated solution for service composition on top of a Grid environment needs (*i*) advanced support for the design and analysis of compound services, (*ii*) their decentralized execution in a reliable way, and (*iii*) sophisticated support for service management as it is provided by the IS and the Keeper. In terms of design and verification, it is important to easily combine services by specifying control and data flow and to verify whether proper failure handling is considered. In terms of execution, the functionality of the process engine should be distributed in the Grid. Despite of the lack of a centralized process engine which would quickly become a bottleneck when the number of processes increases, sophisticated failure handling strategies have to be applied and enforced.

4.1 Compound Services: Design and Validation

Technologies and standards like XML, SOAP[7], and WSDL[8] provide a simple means to describe services and to make them accessible to a large community in a distributed environment such as a virtual organization cooperating in an eScience DL. Yet, the full potential of web services becomes only apparent if several service invocations can be combined to establish even more powerful composite services (processes). A process defines the logical dependencies between independent services by specifying an invocation order (control flow) as well as rules for the transfer of data items between different invocations (data flow). In addition, it is possible to define the transactional behavior and execution guarantees to ensure a correct execution of processes even in case of concurrency and failures. An infrastructure for transactional processes has to support all these runtime semantics. Furthermore, a graphical process modeling tool should support DL users in specifying their applications.

The modeling tool implemented for DILIGENT for the purpose of *Process Design and Verification* is meant to help users seamlessly build processes. It allows for the easy composition of services into process definitions in a controlled manner. This tool is implemented in Java and runs as an applet, which will be integrated into a portlet running on the DILIGENT portal host.

[7] Simple Object Access Protocol, `www.w3.org/TR/soap/`
[8] Web Service Description Language, `http://www.w3.org/TR/wsdl`

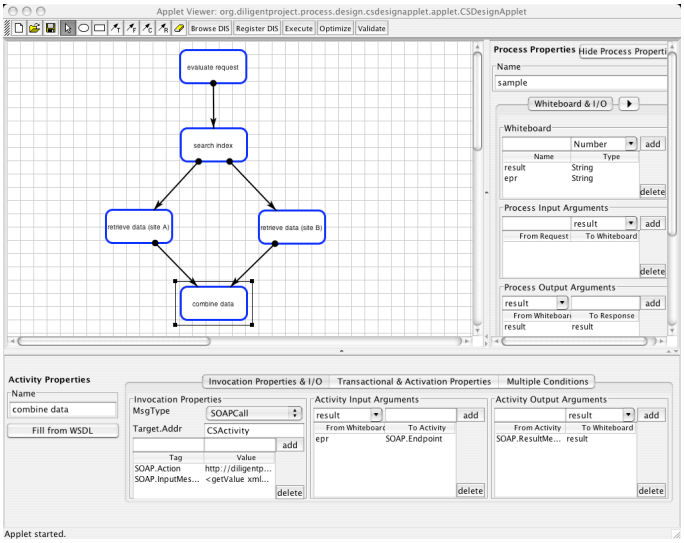

Fig. 4. A Screen Shot of the DILIGENT Modelling Tool for Compound Services

Figure 4 shows a screen shot of it. The modelling tool offers process design in a graphical boxes and arrows approach for the control flow. The rounded boxes in Figure 4 indicate the activities of the process. Each activity corresponds to an invocation of a service. This can either be a basic (web) service or again a process. Furthermore, the data flow of a process is designed by the concept of a process whiteboard, the in-process variable area. At the end of the process design task, the tool outputs the process description and new/modified versions of a process can be registered and uploaded in the DILIGENT Information Service. On the set of activities, two different orders are defined. The partial precedence order specifies the regular order in which services associated with process activities are invoked. An activity can only be executed when all its pre-ordered activities have successfully finished and when the conditions on its execution are fulfilled. Since the precedence order is a partial order, intra-process parallelization can be realized by parallel branches (fork/join). In addition to the precedence order, the preference order specifies alternative executions that can be chosen when an execution path fails. Each activity is described by a set of properties like execution costs, compensation, retriability, and failure probabilities. Given a transactional process description, the tool is able to check correctness of the description based on formal criteria at design time according to the model of transactional processes [15].

In DILIGENT, we are using the BPEL[9] standard as a starting point for the process specification. The DILIGENT modelling tool is generating a BPEL-compliant process definition, which we have enriched with transactional properties.

The Process Design and Verification service is responsible for providing a user interface for viewing, editing, and managing process definitions and for validating

[9] Business Process Execution Language for Web Services, `www.ibm.com/developerworks/library/specification/ws-bpel/`

process definitions defined by a user or generated by another DILIGENT service. The process validation is necessary if the process will be saved in the DILIGENT IS and involves syntactical and semantical correctness checks. The result of the validation process is a "validation signature" confirming that the process is correctly defined. This is required since processes without the validation's signature or whose signature is corrupted will be rejected by the runtime engine.

4.2 Compound Services: Execution

After a process has been designed and stored in the DILIGENT IS, users or other services within the DILIGENT infrastructure may start its execution by sending a *Start CS* message to (any running instance of) the service responsible for the execution of compound services. The process is then executed in a decentralized manner as described below.

Consider the process shown in Figure 5 which corresponds to the sample process from Section 1 (retrieve and combine data from different sources in the earth observation domain). The shapes represent different types of services available in the DILIGENT infrastructure. It should be stressed that the activities specified in the compound service specification refer only to the *type* of the service, not to actual running instances identified by concrete endpoint references. The actual nodes where the process is executed are determined at runtime using information about the deployment of the different services and the current status of the available nodes.

The orchestration and coordination of the execution of a process must be handled by some process execution engine; however, in order to provide true decentralization for the execution, the orchestration of the process execution should not be done by a single instance of such an engine – which would create a single point of failure and a potential hot spot –, but should rather be distributed as well, making use of the available resources and ideally closely following the path that the process execution takes. This implies that the nodes hosting the services to be executed are equipped with the execution engine, so that each activity invoked in a workflow can be handled by the local engine.[10] Therefore, on each node in the DILIGENT infrastructure, a copy of the process execution engine is deployed.

The functionality for coordinating compound service execution is provided by a WSRF service named *CSEngine*. The CSEngine is built on top of the existing distributed process execution engine *OSIRIS* [16,17], which has been modified and enhanced to support WSRF service calls and to use the Information Service provided by the DILIGENT infrastructure. The actual execution of a process is a chain of interaction between the CSEngine services on the involved nodes; each CSEngine in turn invokes the target services locally. Figure 6 gives an overview of the most important components of the CSEngine service and the interactions taking place during the execution of a process (in this case, the execution of activity 3 of the process depicted in Figure 5).

The main functionality of the *WSRF Service Interface* is to provide the SOAP interface to the CSEngine, i.e., it provides web service operations which forward the received data to the OSIRIS core engine. It also contains methods for invoking these

[10] It is not absolutely required that each node have the execution engine installed, since web service calls can of course be done remotely; however it is advisable.

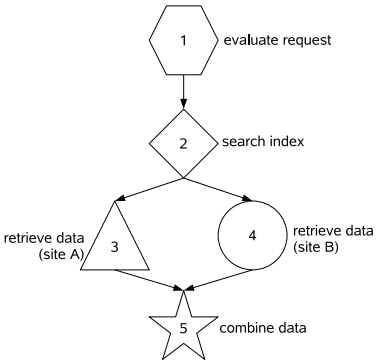

Fig. 5. Sample Process

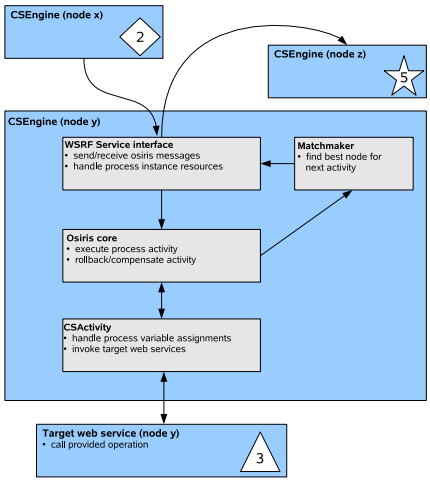

Fig. 6. Architecture of the CSEngine Service

operations on other CSEngine nodes, i.e., for forwarding process execution messages. Finally, this component also maintains the process state, as a WS-Resource, for each compound service execution the node has been involved in.

The *OSIRIS Core* component handles the orchestration of the execution of activities. Given a message, as received by the WSRF service interface, and the process definition (replicated from the DILIGENT IS), it determines which step of which process this message corresponds to. It enriches the message with all additional required information and forwards the message to the relevant component (the CSActivity component in DILIGENT). On return, the process definition is checked for the successor activities of the current node, the corresponding OSIRIS messages are generated and ultimately forwarded to the Matchmaker.

CSActivity is the component which actually executes the given compound service activity. At the moment, two types of activities are implemented, which correspond to the BPEL `<invoke>` and `<assign>` activities; the former invokes a target web service (typically residing on the same node as the CSEngine), the latter is used for evaluating and performing process variable assignments. In the future, additional activity types will be added, in order to support more BPEL constructs (e.g., loops, switches etc.)

Finally, the *Matchmaker* receives an OSIRIS message containing, among others, information about the next activity to be executed in the compound service. Given this message (in particular the type of activity and the type of the target web service), the matchmaker is in charge of determining the best node for handling this message, as described below. The resulting target node is added to the message, and the message is handed back to the WSRF Service Interface for forwarding to the CSEngine on the target node.

The most important aspect of process execution in DILIGENT is its distribution: even orchestration of the process execution is distributed among multiple instances of the *CSEngine* service. At runtime, ideally the only remote service calls take place

between the involved CSEngine instances, while the invocations of the actual target services for a process activity are handled locally at the respective target node.

The decision where the next activity of a process is to be executed is made dynamically at runtime by the Matchmaker using information from the DILIGENT IS. On startup, the component gathers a list of all services running on the local node. It then queries the IS for process specifications containing invocations of either of these services, because the CSEngine may become involved in the execution of these processes. Finally, a list of potential successor services (i.e., the *next* activities in the respective processes). The IS is then queried for deployment information about these services, and for QoS-related information about the corresponding nodes – a prominent example would be the current load of the node. With all this information available, the routing decision can be made using local knowledge only. However, the requested information may change over time (e.g., new process specifications may be added, service instances may be deployed or undeployed, the state of the nodes will change, etc.). Therefore, corresponding subscriptions to notifications about status changes are set up. This allows for timely updates of the state information, while avoiding the overhead which would be introduced by constant polling.

5 Exploitation: the ImpECt Scenario

The DILIGENT infrastructure is evaluated in different application domains which cover the full spectrum from rather traditional Digital Libraries in cultural heritage applications to complex and dynamic e-Science DLs.

In particular, the ImpECt (Implementation of Environmental Conventions) scenario which includes leading actors from the environmental sector and which is represented in the DILIGENT consortium by the European Space Agency (ESA) comes along with challenging requirements for applications from the earth observation domain. These include dynamic and on demand deployment of services as well as process definition, verification, and execution. In this framework, traditional DLs are not suitable anymore to satisfy emerging activities like, for example, assessments and planning responses to environmental accidents. International and regional conventions related to marine pollution and the UNESCO World Heritage Programme represent the framework for formulating international environmental agreements. To face such situations DLs must provide facilities for storing, managing and accessing multi-type information, for making community specific applications available, and for responding to proper on-demand person-centric aggregation and interoperability of data and services within user-defined processes. These above mentioned conventions are continuously evolving and thematic areas are specialising. Yet, information sources are dispersed among environmental agencies and a DILIGENT-based DL could be the most appropriate tool for enabling this community to more effectively coordinate actions.

The Earth Observation tasks of ESA are mainly centered on the combination and correlation, the evaluation and the visualization of many different data stemming from different sources. In order to facilitate the integration, the data as well as the operations for working with the data are made available through web services. The involvement of the European Space Agency (ESA) as a user community in the DILIGENT project strives to make use of the DL infrastructure for complex search

and data processing tasks. The latter demand for the flexible and dynamic utilization of services in the Grid and their combination in order to support the complex data production and dissemination chain. At the same time, users have to be provided with an easy-to-use interface.

Consider the following example: to answer questions about the correlation between the air pollution and the proliferation of algae in the Mediterranean sea over the last two years, one might wish to automatically generate a report which retrieves and combines the relevant satellite images and measurement data, and finally generates a document containing the combined images and corresponding derived metadata in chronological order. Clearly, this task corresponds to a composite service, consisting at least of the following basic services: (*i*) evaluating the request, (*ii*) searching for relevant documents and images according to the filter criteria, (*iii*) retrieving the data, (*iv*) possibly transforming the data into a unified format, (*v*) combining the data (e.g., to create an overlay image, (*vi*) generating derived data, and finally (*vii*) integrating the data into a document.

The process management components of the DILIGENT testbed will be used for the definition and execution of the corresponding processes. These, in turn, rely on the Information Service and the Keeper for managing and deploying the earth observation services on demand.

6 Conclusion

Current IT infrastructures have to take into account that research is more and more becoming a cooperative and multidisciplinary effort. Supporting virtual research organisations requires the possibility to create powerful environments for collaboration, cooperation and for sharing and exploitation of data by dynamically making use of existing resources. For Digital Libraries, this means that they are no longer static systems providing access to well-defined collections. Rather, e-Science DLs have to be dynamically tailored to particular applications needs. This includes the provision of access to several information sources, the support of complex information production cycles comprising the creation, annotation, exploration, dissemination, and exchange of data. Therefore, two requirements are vital for e-Science DLs. First, the support for the on-demand adaptation to changes in the DL applications but also to changes in the underlying environment . Second, the support for complex applications that require the combination of different resources which are usually made available by means of services. A key to meeting this requirements is the use of *Grid* infrastructures as the basis for advanced e-Science DLs.

In this paper, we have presented the DILIGENT approach to providing on-demand adaptation and workflow support in e-Science DLs. These two aspects are part of a comprehensive infrastructure for powerful, dynamic DLs on top of a Grid environment. In addition, current activities in DILIGENT also exploit the underlying Grid infrastructure for storage purposes (of individual information objects, their associated metadata, and collections), and for sophisticated search (e.g., by making use of the Grid for the parallelization of complex feature extraction). Another main line of activities in DILIGENT is the evaluation of the infrastructure in concrete application scenarios. For the dynamic adaptation and process support, the most challenging one is the ImpECt scenario driven by the ESA since they have many

complex processes to support for data analysis and a large number of (computationally intensive) services available. However, since DILIGENT is designed to be an application-independent infrastructure, evaluation will consider other scenarios from different application domains as well.

References

1. W. Y. Arms. *Digital Libraries*. The MIT Press, September 2001.
2. T. Banks. Web Services Resource Framework (WSRF) - Primer. Committee draft 01, OASIS, December 2005. `http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-01.pdf`.
3. DILIGENT. A DIgital Library Infrastructure on Grid ENabled Technology. `http://www.diligentproject.org/`. IST-2003-004260.
4. EGEE. Enabling Grids for E-sciencE. `http://public.eu-egee.org/`. INFSO 508833.
5. EGEE. gLite: Lightweight Middleware for Grid Computing. `http://glite.web.cern.ch/glite/`.
6. I. Foster. What is the Grid? A Three Point Checklist. *GRIDtoday*, 1(6), 2002.
7. I. Foster and C. Kesselman. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan-Kaufmann, 2004.
8. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.
9. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organization. *The International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.
10. E. A. Fox, R. M. Akscyn, R. Furuta, and J. J. Leggett. Digital Libraries. *Communications of the ACM*, 38(4):23–28, April 1995.
11. E. A. Fox and G. Marchionini. Toward a Worldwide Digital Library. *Communications of the ACM*, 41(4):29–32, April 1998.
12. GÉANT Team. GÉANT Web Site. `http://www.geant.net`.
13. Globus Alliance. The Globus Alliance Website. `http://www.globus.org/`.
14. Y. Ioannidis. Digital libraries at a crossroads. *International Journal on Digital Libraries*, 5(4):255–265, August 2005.
15. H. Schuldt, G. Alonso, C. Beeri, and H.-J. Schek. Atomicity and Isolation for Transactional Processes. *ACM Transactions on Database Systems (TODS)*, 27(1):63–116, Mar. 2002.
16. C. Schuler, H. Schuldt, C. Türker, R. Weber, and H.-J. Schek. Peer-to-Peer Execution of (Transactional) Processes. *International Journal of Cooperative Information Systems (IJCIS)*, 14(4):377–405, 2005.
17. C. Schuler, C. Türker, H.-J. Schek, R. Weber, and H. Schuldt. Scalable Peer-to-Peer Process Management. *International Journal of Business Process Integration and Management (IJBPIM)*, 1(2):129–142, 2006.
18. B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, and R. Wolski. A Grid Monitoring Architecture. Technical Report GFD.6, Global Grid Forum Document Series, 2002.