Towards an Integration Platform for Bioinformatics Services

Guzmán Llambías, Laura González, and Raúl Ruggia

Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay
{gllambi,lauragon,ruggia}@fing.edu.uy

Abstract. Performing in-silico experiments, which involves an intensive access to distributed services and information resources through Internet, is nowadays one of the main activities in Bioinformatics. Although existing tools facilitate the implementation of workflow-oriented applications, they lack of capabilities to integrate services beyond low-scale applications, particularly integrating services with heterogeneous interaction patterns and in a larger scale, ideally based on a Platform as a Service paradigm. On the other hand, such integration mechanisms are provided by middleware products like Enterprise Service Buses (ESB). This paper proposes an integration platform, based on enterprise middleware, to integrate Bioinformatics services. It presents a multi-level reference architecture and focuses on ESB-based mechanisms to provide asynchronous communications, event-based interactions and data transformation capabilities.

Keywords: Platform as a Service (PaaS), Scientific Platforms, Middleware.

1 Introduction

An in-silico experiment is a procedure that uses computer-based resources (local and remote) to test a hypothesis, derive a summary or search for patterns [1]. Bioinformaticians develop these experiments using Workflow Management System tools, specially adapted to the biological context, giving birth to scientific workflows.

Scientific workflow tools, notably Taverna [2], have revolutionized the way researchers perform experiments by enabling them to use powerful computational tools without needing a strong IT background. These tools enable to access external services via Web Services, perform data format transformations, execute queries on large databases and even request the execution of an experiment in the cloud [3]. Although the generalized use of Taverna is a clear success indicator [2], it does not provide suitable mechanisms to handle some particular characteristics of bioinformatics services [4] (e.g. the polling approach followed by most biological service providers and the use of different data formats in services). This fact turns the logic to be implemented by workflows more complex. Furthermore, it doesn't provide asynchronous message-oriented mechanisms and quality of service management.

On the other side, middleware technologies which have been evolving during the last years, provide abstractions and solutions to increasingly complex integration issues (e.g. asynchronous communications and interoperable communications over

A.R. Lomuscio et al. (Eds.): ICSOC 2013 Workshops, LNCS 8377, pp. 445–456, 2014.
© Springer International Publishing Switzerland 2014

the internet) related to the construction and integration of distributed applications. In particular, Enterprise Service Buses (ESB) and cloud-based Internet Service Buses (ISB) [5], are sophisticated middleware technologies which enable to integrate highly distributed and heterogeneous services. ESBs provide rich mediation capabilities (e.g. message transformation and intermediate routing) which can be used to address mismatches between applications and services (e.g. regarding communication protocols, message formats, interaction styles and quality of service) [6] [7].

This leads to the challenge of improving collaboration in bioinformatics community by enhancing service-based integration capabilities and reducing the complexity in the involved development. Such enhanced platforms should enable to integrate bioinformatics Web-based services (e.g. NCBI Web Services) and experiment-oriented workflow tools (e.g. Taverna) with general purpose mediation features and other value-added capabilities. The ultimate goal is to put into practice a Platform as a Service (PaaS) approach in the bioinformatics area, enabling an active participation of laboratories, researchers, and middleware and cloud computing suppliers and developers.

This paper addresses these issues and proposes a reference integration platform for the bioinformatics domain which, mediating between Taverna (Kepler[1], Galaxy[2], etc.) and bioinformatics services, provide mechanisms that facilitate the development of distributed scientific workflows and solve common challenges arising when performing this task. The proposed integration platform leverages mediation features of enterprise middleware (particularly ESBs), addresses identified integration requirements by improving asynchronous interactions, event notification and message transformation capabilities, and provide the means to implement other value-added services.

This paper, which is part of a larger work jointly developed with the Bioinformatics Unit of the Pasteur Institute at Montevideo, focuses on solutions oriented to medium and large-scale integration platforms potentially involving cloud resources.

The rest of the paper is organized as follows. Section 2 presents background. Section 3 presents a high level view of the proposed solution. Section 4 describes an ESB-based design which provides solutions for some identified scenarios. Section 5 presents related work and finally, Section 6 presents conclusions and future work.

2 Background

2.1 Cloud Computing and PaaS

According to NIST [8], *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction"*.

On the other side, Platform as a Service (PaaS) is one of the five service models that compose Cloud Computing. It provides the consumer the capability to deploy, in the cloud infrastructure, applications built on top of the elements (e.g. programming

[1] `https://kepler-project.org/`
[2] `http://galaxyproject.org/`

languages) supported by the provider. The consumer does not manage or control the underlying cloud infrastructure, but he has control over the applications he deploys and over configuration settings for the hosting environment [8].

2.2 Taverna and myGrid Project

myGrid[3] is an e-science middleware project, whose main purpose is to provide middleware based tools of high abstraction that can simplify the development of bioinformatics experiments. One of the tools developed in this context is Taverna: a tool that allows scientists to model their experiments as a composition of biological services, building in this way scientific workflows. The main contribution of Taverna is that it allows scientists with limited IT expertise, to be capable of developing their experiments using advanced technologies such as Web Services, BioMart, R, etc [2].

2.3 Enterprise Service Bus (ESB)

An ESB is an environment belonging to the platform middleware systems category, which provides sophisticated interconnectivity between services and enables to overcome issues related to reliability, scalability and communications. Service interaction using an ESB is based on a combination of the patterns: Asynchronous Queuing, Event-Driven Messaging, Intermediate Routing, Policy Centralization, Reliable Messaging, Rules Centralization and Transformation [9]. Additionally, interaction styles define the way each actor may behave using an ESB.

ESB Design Patterns. *Intermediate routing patterns* dynamically determine the message path according to different factors. More concretely, a content-based router defines the message path based on its content and a recipient list routes the message to a list of dynamically specified recipients.

Transformation patterns deal with the runtime transformation of messages. In [9] the authors identify three types of transformations: data model transformation, data format transformation and protocol bridging.

The *Asynchronous Queuing pattern* deals with the interactions of client and services when synchronous communication can affect performance and reliability.

Interaction Styles. In [11], eight ESB interaction patterns are identified. These patterns are abstract and define the way actors behave and interact in terms of synchronous/asynchronous interfaces, level of assurance of delivery, handling of timeouts, late responses, and error handling. In this paper, we focus on two interactions styles: 1) Synchronous update Request with acknowledgement and callback and 2) One-way with status poll for completion.

[3] `http://www.mygrid.org.uk/`

3 Towards a Bioinformatics Integration Platform

An environment for in-silico bioinformatic experiments includes, on one hand, local bioinformatic tools (e.g. *Taverna*) which enable researchers to implement experiments by composing resources. On the other hand, there are bioinformatic resources, especially external and distributed over the Internet (e.g. NCBI, EBI, DDBJ), which are accessible via APIs, Web Services SOAP and REST.

Although bioinformatic tools like *Taverna* provide user oriented development features, their lack of mediation mechanisms (e.g. asynchronous interactions, event management, declarative data transformations) and management functionalities (e.g. quality of service, service policy) limits their ability to integrate all the involved elements.

The proposed Bioinformatics Integration Platform provides features to implement an advanced interaction between bioinformatics applications and services (Fig. 1). Concretely, it enables to perform not only traditional synchronic function invocations, but also asynchronous interactions based on a request-response pattern and data transformation mechanisms.

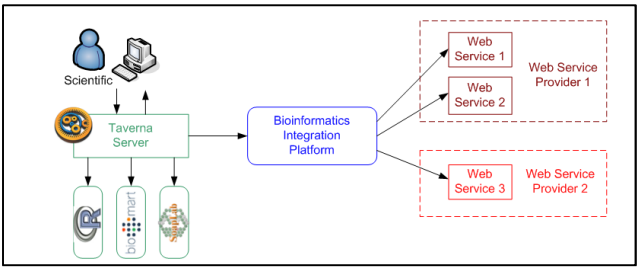

Fig. 1. The Bioinformatics Integration Platform in Context

The specification of the integration platform consists of several refinement levels. The highest level, shown in Fig. 1, is independent from technical approaches and laboratory application contexts. Specifications in a subsequent level are based on the technical approaches followed to solve integration requirements in different laboratory scenarios by using specific middleware technologies. The third refinement level introduces middleware product aspects related to specific implementations.

The following parts of this section focus on these aspects, starting with solutions proposed to integration requirements (asynchronous interaction, events and notification, and data transformation) and following with a classification of Laboratory scenarios characterized by their scale and service capabilities. The overall architecture, based on the multilayer refinement specification, is presented at the end of the section.

3.1 Integration Requirements

Asynchronous Interactions. Nowadays, many biological service providers design their Web Services using asynchronous communications due to the large amount of

time and resources they use to perform data processing. After studying some Web Services of the NCBI[456], we can argue that they were designed using a response polling model, where clients send a job request to the Web Service and receive a jobId, to use afterwards and check for its status (waiting, running, finished). Only when the job status is finished, the client can poll the response. Fig. 2 presents this model.

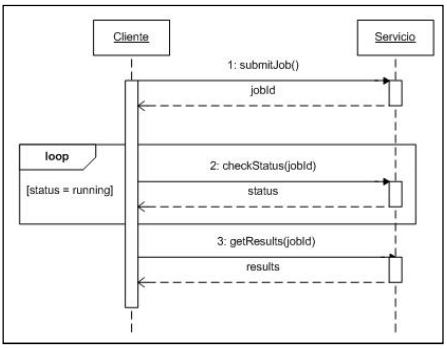

Fig. 2. Polling based asynchronous communication.

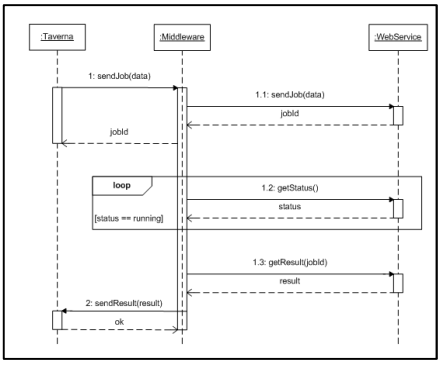

Fig. 3. Callback based asynchronous communication

Although the polling model is very effective, it is not too efficient as it requires three tasks and a loop to perform only one analysis. A callback approach as showed in Fig. 3 is much more efficient and equally effective as the polling model. This paper proposes that the design of asynchronous bioinformatics Web Services use a callback approach instead of the polling model. If this is not possible, middleware technologies should be used to mediate between client and service to reach this approach.

Events and Notifications. The completion of a workflow, the receipt of a message from an external system or a timeout to finish a task, are different types of events which are provided as natives features in many enterprise workflow management tools (WS-BPEL, Windows Workflow Foundation, JBPM, etc). As far as we know, Taverna does not have support for these features, and it has limited support for basic event notifications (Atom feeds, email, SMS, Twitter and Jabber when workflows are completed). The latter are very useful for human notification but are not suitable for machine to machine notifications or the integration of Taverna to other information systems. Some useful examples of notifications are Database updates (each update of the Genbank database could be notified to Taverna to rerun existing workflows), Receive notifications of available information from an external system (i.e. receive or

[4] http://www.myexperiment.org/workflows/203.html
[5] http://www.myexperiment.org/workflows/210.html
[6] http://www.myexperiment.org/workflows/230.html

wait WS-BPEL activities) and Workflow completion (notify external systems by more sophisticated means other than the ones supported nowadays).

This paper proposes the application of middleware-based technologies to provide a notification mechanism based on the Publish/Subscribe design pattern, which could allow Taverna to subscribe to events and notifications from external systems.

Transformation Services. As scientific workflows are usually based on the composition of third party services, scientists have to transform the output format of one service to the input format of the next service (e.g. transform a plain text output into a FASTA format input). A survey of 415 registered workflows in myExperiment found that 30% of the tasks involved in each experiment were due to format changes, while just 22% were due to task for invoking Web Services [12]. Therefore, shim services are needed to accomplish the parsing and transformation of data formats [13], but this task is hardened by the lack of accepted standards and data models. BioXSD [14] and PhyloXML [15] are two examples of standardization initiatives for sequence alignment and phylogenetic data, with yet, low acceptance by the scientific community.

In [14], three possible scenarios are analysed based on how much alignment exists between the input and output data formats. In the first scenario, services receive and return data in plain text using the data type xsd:string. It is necessary to use shim services to transform data formats. In the second scenario, services return and receive data in xml format but the output data model is different from the input model of each service. It is necessary to use shim services or xslt scripts to perform the transformations. Finally, in the third scenario, services use the same data model and xml format. There is no need for transformations and data flows smoothly from one service to the other.

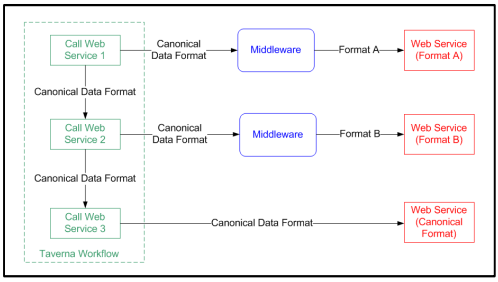

Fig. 4. Middleware-based message transformations scenarios

It would be desirable to reduce as much as possible the burden of shim services and try to seek for C scenarios, where there is no need for data parsing and transformations. Today, part of this objective can be achieved if the services use canonical data formats based on international standards. However, much work has to be done in this sense and in the meanwhile, alternative solutions are needed to achieve this goal.

This paper proposes to include intermediaries that provide standard service views to clients. The intermediate broker will offer the same service to clients but in a suitable data format performing the necessary transformations (Fig. 4).

3.2 Laboratory Scenarios

The characteristics of the Bioinformatics Laboratories, especially the ability to export new services, influences the way in which the Bioinformatics integration platform may be used. The following classification enables to instantiate this aspect:

- Laboratory type 1: Uses Taverna as a standalone tool, using the native connectors to consume biological services (WSDL, SoapLab, R).
- Laboratory type 2: Uses Taverna to consume biological services using the native connector, while uses point-to-point middleware to consume services not supported natively by Taverna. The number of middleware-based integrations is low.
- Laboratory type 3: Uses Taverna to consume compatible biological services while uses a platform middleware to consume other services not compliant with native Taverna integration. This type of laboratory is a large-scale service consumer.
- Laboratory type 4: Provides services to other laboratories while not necessarily consumes. It focuses on interoperability and accessibility features. Uses point-to-point or platform middleware depending on the number of provided services.
- Laboratory type 5: Provides Platforms as a Service (PaaS) and it doesn't focus on biological business aspects. One example of this type of Platforms is the Amazon SWF. This type of laboratories uses platform middleware.

3.3 Overall Architecture

Globally, the Bioinformatics Integration Platform can be defined as a domain-specific middleware, which provides abstractions and high quality services to solve common integration requirements in a bioinformatics context. Its architecture, shown in Fig. 1 at the highest abstraction level, can be refined by instantiating three dimensions:

- Integration functionalities: asynchronous interaction, events and notification, and data transformation.
- Laboratory scenarios: Laboratory types from 1 to 5.
- Middleware technologies: Web Services, Message Queues, ESB & ISB.

Characterizing solutions through these dimensions enables to specify and implement them, keeping the consistency with the overall architecture and with other differently instantiated solutions. Notably, Fig. 5 shows connections between specific architectures for laboratory scenarios (type 1 to type 3) based on different middleware technologies. In section 4, specific solutions are described for laboratories scenarios type 4 and type 5, using ESB as middleware technology.

4 An ESB-Based Bioinformatics Integration Platform

The Bioinformatics Integration Platform can be designed and implemented using various middleware technologies (Queues, Web Services, ESBs, etc). This section presents an ESB-based Bioinformatics Integration Platform, which is suitable for Platform as a Service (PaaS) development (Laboratory type 5 scenario), and shows how the requirements of Section 3 are supported.

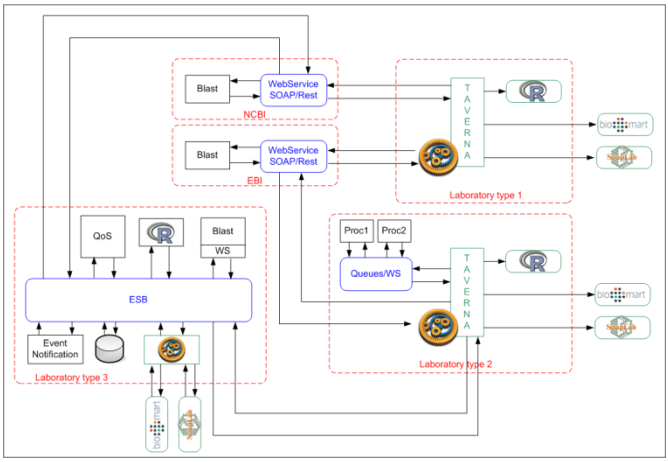

Fig. 5. Overall architecture instantiated in different laboratories

4.1 Asynchronous Communications

The proposed solution (Fig. 6) is ESB-based and follows the patterns: Asynchronous Queuing (i.e. ESB message queues), Intermediate Routing (i.e. Content Based Router, CBR) and Protocol Bridging (i.e. ESB Endpoints and ESB Connectors) [9].

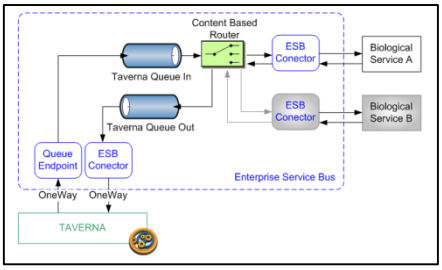

Fig. 6. Asynchronous communications based on ESB

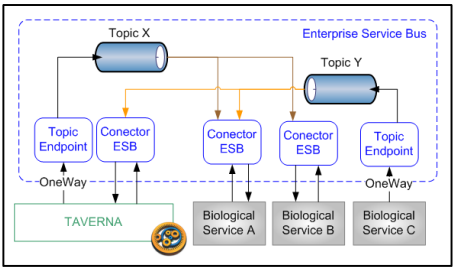

Fig. 7. Events and notifications based on ESB middleware

This solution involves defining two message queues in the ESB: one used to receive messages from Taverna (TavernaQueueIn) and the other to send the responses (TavernaQueueOut) from the ESB to Taverna. The only subscriber of the TavernaQueueIn message queue is a CBR, which inspects and routes messages to the biological service through a specific ESB Connector, and uses messages' content to identify the destination service. All request messages must specify a destination service, which should be registered in the Platform's Service Catalog. After processing the request, the service sends a response message to the TavernaQueueOut queue, whose only subscriber (ESB Connector) forwards the message to Taverna. Since there may be multiple instances of Taverna waiting for response messages, a replyTo attribute must be defined in the request messages to identify the destination instance.

4.2 Events and Notifications

The proposed solution is based on an ESB-oriented design and takes advantage of the Publish&Subscribe and Protocol Bridging (i.e. Topic Endpoints and ESB Connectors) design patterns [9]. This solution defines a topic in the ESB for each type of event or notification to be managed by the Platform. Each topic is identified by a TopicID and may have N publishers and M subscribers, where each subscriber may subscribe using two different types: synchronous or asynchronous.

To notify an event to the Platform, the publisher creates a message with business data and a TopicID, and sends it to the Endpoint Topic in the ESB. This component places the message in the corresponding topic according to the TopicID and acknowledges its reception. Each subscriber of the topic will receive a copy of the message according to the chosen subscription type. For asynchronous subscribers, the topic sends a copy to the ESB Connector associated with the service according to the communication protocols and specific data format. Synchronous subscribers instead, should query the topic through the Endpoint Topic for new messages.

4.3 Transformation Service

The design of this solution (Fig. 8) is ESB-based and applies the Content Based Routing, Canonical Data Model and Protocol Bridging (i.e. ESB Connectors and ESB Endpoints) design patterns [9]. The solution consists in automatically determining the format of the messages sent by Taverna and transforming them to the data format required by the service. To reduce the number of transformations to be configured, the Canonical Data Model pattern is applied, transforming incoming messages to a canonical data model and later-on back to the specific service data format. A Content Based Router is used to route incoming messages to the service. Native connectors are used (ESB Connectors) to perform the communication between the ESB and services.

4.4 Implementation Details

To show the feasibility of the proposed approach and to analyse key implementation aspects, prototypes were developed using JBoss ESB and NCBIs Web Services.

Concretely, Asynchronous Communications were implemented using JBoss's built-in features: ESB Queues and the CBR. Events and notification requirement were implemented using ESB's Topics. The transformation requirement was developed using the JBoss's CBR and XSLTs engines. The development of XSLT style sheets were needed to be used by the message transformation engine. The ESB Connectors used in all the scenarios were provided by JBoss's SOAP Proxy. Finally, the ESB Endpoint was also a JBoss's JMS Listener and HTTP Gateway.

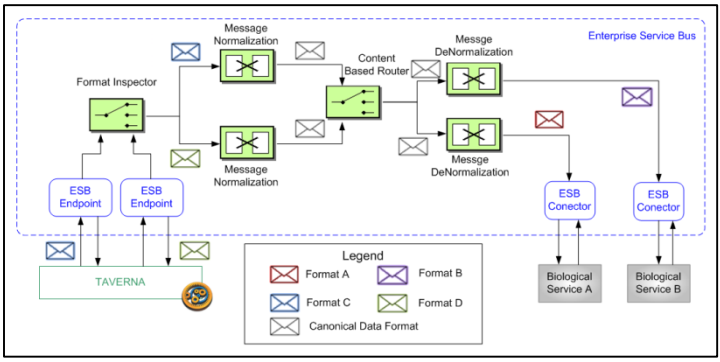

Fig. 8. Data/Model transformation based on ESB

5 Related Work

While different proposals have addressed asynchronous communications issues in scientific workflows, they present limitations. Particularly, [16][17] assume that biological Web Services have WS-Addressing support while this is not frequent (e.g. EBI, NCBI, DDBJ do not provide it). Our approach does not depend on this feature.

Notification mechanisms and events have also been addressed in Bioinformatics. In [17] Apache ODE, a WS-BPEL engine, is extended to send notifications in an e-science context. This solution has the limitation of being too coupled to Apache ODE. In [18] a Web Services based message broker is proposed to send notifications. The main difference with our approach is the scale of the solution: while this broker is lightweight and suitable to be integrated into other platforms, our mechanism is part of a wider integration platform for Bioinformatics built on top of an ESB. In [19] a WS-Eventing middleware is proposed to improve interoperability between workflows tools. This solution only allows synchronous subscriptions while ours also supports asynchronous ones. In [20] a notification bus is proposed integrating all the components. Unlike this proposal, our solution is based on a service oriented design and leverages an existing ESB infrastructure.

Regarding format transformation, the excessive use of shim services is addressed by automatically identifying which shim services are required to be included in the workflow [13]. Our work proposes reducing as much as possible, their use. In [21] the concept of Virtual Data Assembly Lines (VDAL) is presented to hide the use of shim

services to scientists. While VDALs are locally defined in each workflow, our solution provides a global view of the transformed service to the whole organization.

Unlike the previously described work, our approach aims to provide a comprehensive domain-specific platform for bioinformatic services. This kind of platform provides building blocks and services adapted to a specific domain. The Open eHealth Integration Platform [22] is a platform middleware for e-health aiming at providing mechanisms to integrate e-health applications (e.g. HL7 message processing). In [23] an ESB-based platform is proposed in the context of geographic information systems. It uses mediation mechanisms (e.g. SOAP WMS-Wrapper) to facilitate the integration of geographic Web Services and enterprise applications. To the best of our knowledge there are not proposals of domain-specific integration platforms in the Bioinformatics domain using enterprise middleware technologies.

6 Conclusions and Future Work

This paper addresses the issues of improving Bioinformatics laboratory collaboration and proposes a reference integration platform which provides enhanced capabilities to implement distributed and service-based systems.

The implementation approach, based on enterprise middleware technologies (ESB, etc.), has shown to be capable of addressing the requirements of providing advanced integration features and adequately connecting to Taverna and other Bioinformatics services. Still, functionalities like processing very large data sets require further work.

The main contributions of this work consist in the analysis and identification of relevant features to be provided in a Bioinformatics integration platform, the proposed solution that can be applied to different types of laboratories, and the implementation of prototypes that enabled to validate technologies and the implementation approach.

The work also constitutes a step forward on carrying out a Platform as a Service (PaaS) approach for Bioinformatics.

Future work consists in extending the presented results with service composition and policy rules management mechanisms. On the other hand, it could include the conceptualization of Domain Specific integration platforms, beyond the Bioinformatics context, based on a PaaS approach as well as on the here applied framework.

References

1. Stevens, R., Glover, K., Greenhalgh, C., Jennings, C., Pearce, S., Li, P., Radenkovic, M., Wipat, A.: Performing in silico Experiments on the Grid: A Users' Perspective (2003)
2. Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., SoilandReyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hard-isty, A., Nieva de la Hidalga, A., Balcazar Vargas, M.P., Sufi, S., Goble, C.: The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. Nucleic Acids Research 41, W557–W561 (2013)
3. Dou, L., Zinn, D., McPhillips, T.M., Köhler, S., Riddle, S., Bowers, S., Ludäscher, B.: Scientific workflow design 2.0: Demonstrating streaming data collections in Kepler, ICDE 2011, pp. 1296–1299 (2011)

4. Llambías, G., Ruggia, R.: Taverna: un ambiente para el desarrollo experimentos científicos. Pedeciba Informatica
5. Ferguson, D.F.: The Internet Service Bus. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, p. 5. Springer, Heidelberg (2007)
6. Schmidt, M.-T., Hutchison, B., Lambros, P., Phippen, R.: The enterprise service bus: making service-oriented architecture real. IBM Syst. J. 44, 781–797 (2005)
7. Wiederhold, G.: Mediators in the architecture of future information systems. Computer 25, 38–49 (1992)
8. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. NIST (2011)
9. Erl, T.: SOA design patterns. Prentice Hall, Upper Saddle River (2009)
10. Hohpe, G.: Enterprise integration patterns: designing, building, and deploying messaging solutions [..] [..]. Addison-Wesley, Boston (2003)
11. Enterprise Connectivity Patterns: Implementing integration solutions with IBM's Enterprise Service Bus products, http://www.ibm.com/developerworks/library/ws-enterpriseconnectivitypatterns/
12. Wassink, I., van der Vet, P.E., Wolstencroft, K., Neerincx, P.B.T., Roos, M., Rauwerda, H., Breit, T.M.: Analysing Scientific Workflows: Why Workflows Not Only Connect Web Services. (presented at the July 2009)
13. Hull, D., Stevens, R., Lord, P., Wroe, C., Goble, C.: Treating shimantic web syndrome with ontologies. University, Milton Keynes (2004)
14. Kalaš, M., Puntervoll, P., Joseph, A., Bartaševičiūtė, E., Töpfer, A., Venkataraman, P., Pettifer, S., Bryne, J.C., Ison, J., Blanchet, C., Rapacki, K., Jonassen, I.: BioXSD: the common data-exchange format for everyday bioinformatics web services. Bioinformatics 546, i540–i546 (2010)
15. Han, M.V., Zmasek, C.M.: phyloXML: XML for evolutionary biology and comparative genomics. BMC Bioinformatics 10, 356 (2009)
16. Perera, S., Gannon, D.: Enabling Web Service extensions for scientific workflows. In: Workshop on Workflows in Support of Large-Scale Science WORKS 2006, pp. 1–10 (2006)
17. Gunarathne, T., Herath, C., Chinthaka, E., Marru, S.: Experience with adapting a *WS-BPEL* runtime for eScience workflows. In: Proceedings of the 5th Grid Computing Environ-ments Workshop, pp. 1–7. ACM, New York (2009)
18. Huang, Y., Slominski, E., Herath, C., Gannon, D.: Wsmessenger: A web services-based messaging system for service-oriented grid computing. In: CCGrid (2006)
19. Alqaoud, A., Taylor, I., Jones, A.: Publish/subscribe as a model for scientific workflow interoperability. In: Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science, pp. 1:1–1:10. ACM, New York (2009)
20. Gannon, D., Christie, M., Marru, S., Shirasuna, S., Slominski, A.: Programming Paradigms for Scientific Problem Solving Environments. In: Gaffney, P.W., Pool, J.C.T. (eds.) Grid-Based Problem Solving Environments, pp. 3–15. Springer, US (2007)
21. Zinn, D., Bowers, S., McPhillips, T., Ludäscher, B.: Scientific workflow design with data assembly lines. In: Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science, pp. 14:1–14:10. ACM, New York (2009)
22. IPF Overview - Open eHealth Integration Platform 2.x - Confluence, http://www.openehealth.org/display/ipf2/IPF+Overview
23. Rienzi, B., González, L., Ruggia, R.: Towards an ESB-Based Enterprise Integration Platform for Geospatial Web Services. Presented at the GEOProcessing 2013, The Fifth International Conference on Advanced Geographic Information Systems, Applications, and Services (February 24, 2013)