Peer-to-Peer Infrastructure for Multi-Knowledge Workflows

Michele Amoretti, Francesco Zanichelli
Department of Information Engineering
University of Parma, Italy
{amoretti, zanichelli}@ce.unipr.it

Diego Ardigò
Department of Internal Medicine
University of Parma, Italy
ardigo@dim.unipr.it

George Antikatzidis
IMGROUP
London
antikatzidis@imgroup.com

Sergio Copelli, Fabio S. Maresca
R&S INFO Srl
Parma
RSINFO@rsinfo.it

Franco Mercalli
Centro Volta
Como
mercalli@centrovolta.it

Abstract

The general aim of the Multi-Knowledge project is to develop a collaborative environment to allow networks of co-operating medical research centres to create, exchange and manipulate new knowledge from heterogeneous data sources.

The Multi-Knowledge service-oriented architecture (MK-SOA) will enable workflow design and execution based on novel operating procedures to manage and combine heterogeneous data and make them easily available for the imputation of study algorithms. In this paper we focus on the peer-to-peer infrastructure we have developed to support the creation of a fully decentralized collaborative environment, in which each party remains autonomous and the interactions between services are seen from a global perspective.

1. Introduction

The Multi-Knowledge project [4], which is funded by the European Commission, starts from the data processing needs of a network of Medical Research Centres, in Europe and USA, partners in the project and cooperating in researches related to the link between metabolic diseases and cardiovascular risks. These needs are mostly related to the integration of three main sources of information: clinical data, patient-specific genomic and proteomic data (in particular data produced through microarray technology), and demographic data.

In this context the aim of Multi-Knowledge is the development and the validation of a knowledge management environment to allow different groups of researchers, dealing with different sources of data and technological and organisational contexts, to create, exchange and manipulate new knowledge in a seamless way. The ambition is also to create a technological and methodological frame that can easily be extended to include additional sources of data and expertise, and can be applied to wider sectors of medical research.

Critical and difficult issues addressed in the project are the management of data that are heterogeneous in nature (continuous and categorical, with different order of magnitude, different degree of precision, etc.), origin (statistical programs, manual introduction from an operator, etc.), and coming from different data environments (from the clinical setting to the molecular biology lab).

The Multi-Knowledge service-oriented architecture (MK-SOA) will enable workflow design and execution based on novel operating procedures to manage and combine heterogeneous data and make them easily available for the imputation of study algorithms. In this paper we focus on the already-available MK-SOA communication infrastructure, which is based on the peer-to-peer paradigm and provides the glueing middleware for the creation of Multi-Knowledge collaborative environments, in which each party remains autonomous and the interactions between services are seen from a global perspective.

In the first part of the paper, we discuss related work in the field of biomedical distributed services. In the second part, we illustrate the general template for Multi-Knowledge workflows, together with some practical examples. In the third part, we describe the peer-to-peer infrastructure and the technologies we have used to implement it. Finally, we outline some directions for further research and development.

2. Related Works

In the context of clinical services, the European Commission is funding two complementary projects: COCOON [3] and ARTEMIS [1]. COCOON is aimed at setting up a set of regional semantics-based healthcare information infrastructure with the goal of reducing medical errors. ARTEMIS aims to develop a semantic Web Services based interoperability framework for the healthcare domain. Artemis addresses the interoperability problem in healthcare domain in two respects:

- Functional Interoperability, which is the ability of two or more systems to exchange information.
- Semantic Interoperability, which is the ability for information shared by systems to be understood at the level of formally defined domain concepts so that the information is computer processable by the receiving system.

ARTEMIS builds upon a peer-to-peer architecture in order to facilitate the discovery of healthcare Web Services. In Artemis, healthcare institutes are represented as peers. ARTEMIS peers provide interfaces to the healthcare information systems to enable them to discover and consume the Web Services provided by the other peers. In order to facilitate the discovery of the Web Services, OWL-S profiles are used to describe what the service functionality semantics is in the heathcare domain. For example, when a user is looking for a service to admit a patient to a hospital, he should be able to locate such a service through its meaning, independent of what the service is called and in which language.

On the other side, an increasing number of tools and databases in molecular biology and bioinformatics are available as Web Services. However, there is currently no publicly available registry that describes all of these services. The Biological Web Services (BWS) page [2] describes the main services that are available as of March 2006, with appropriate links.

Among the services listed by BWS, GeneCruiser [13] is a Web Service for the annotation of microarray data, developed at the Broad Institute (a research collaboration of MIT, Harvard and its affiliated Hospitals). GeneCruiser allows users to annotate their genomic data by mapping microarray feature identifiers to gene identifiers from databases, such as UniGene, while providing links to web resources, such as the UCSC Genome Browser. It relies on a regularly updated database that retrieves and indexes the mappings between microarray probes and genomic databases. Genes are identified using the Life Sciences Identifier standard.

A more complex example of Web Service-oriented architecture providing transparent access to biomedical applications on distributed computational resources is the National Biomedical Computation Resource (NBCR) [12], which is based in Grid technologies such as Globus Toolkit. NBCR users are allowed to design and execute complex biomedical analysis pipelines or workflows of services.

Compared to these initiatives, the Multi-Knowledge project is a step forward since its objective is the creation of collaborative environments in which many kinds of actors (phisicians, biomedical researchers, etc.) with different responsibilities (*i.e.* providing different services: workflow design, clinical data collection, microarray data entering, data integration, data analysis) participate in the execution of complex experiments, given specified workflow instances. The support for peer-to-peer service sharing in Multi-Knowledge environments is provided by the JXTA-SOAP component [6] (see section 5), which can be compared to WSPeer [10] and WSPDS [8]. The strength of JXTA-SOAP is that it relies on JXTA protocols, which are becoming the de-facto standard for peer-to-peer system design.

3. Introduction to Orchestration and Choreography Models

Workflow at its simplest is the way work gets from start to finish. More specifically, workflow is the operational aspect (*process logic*) of a work procedure: how tasks are structured, who performs them, what their relative order is, how they are synchronized, how information flows (*routing rules*) to support the tasks and how tasks are being tracked. As the dimension of time is considered in workflow, workflow considers "throughput" as a distinct measure. Workflow problems can be modeled (*process definition*) and analyzed using graph-based formalisms like Petri nets. A *process instance*, or job, is a running instance of a process definition. An *activity* is a task that forms one logical step in a process definition. It can be automated or manual. Automation refers to the ability to define scripts and triggers during process operation. A common type of automated activity is deadline handling, which can automatically send a reminder message or trigger an escalation procedure if a work item fails to be completed by a prescribed deadline. In IT, workflow is concerned with the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall goal. People-based processes and rules-based automation processes are becoming more complementary all the time.

Current workflow products define client-server system architectures where the workflow server is monolithic and centralizes critical *orchestration* services such as process management, activity distribution, worklist management, and directory services. While centralizing workflow functionality in one location offers important benefits such as tracking and possibly transactional support for workflow applications, the monolithic architecture of most centralized servers does not address the needs of distributed workflows on a WAN which must necessarily execute in a distributed manner. Emerging *choreography* standards aim to define a notation for expressing the interaction of distributed processes.

The primary difference between orchestration and choreography is scope. An orchestration model provides a scope specifically focusing on the view of one participant. Instead, a choreography model encompasses all parties and their associated interactions giving a global view of the system. Choreography aims at constricting the behaviours of the services involved in the system ruling the exchange of their messages. Choreography is a definition of the rules which govern the exchange among the parties involved in the business activity. Moreover, the state of the activity is distributed among the entities. On the other hand, orchestration allows the design of a central entity (the orchestrator) which carries out a business activity invoking other services. For instance, if there are two services which require to be synchronized, the first has to send the synchronization message to the orchestrator engine which will forward it to the latter. The orchestrator also stores all the state of the activity it is carrying out.

The following section describes Multi-Knowledge workflows and motivates their requirements for choreography support.

4. Multi-Knowledge Workflows

With reference to the definition of workflow introduced in the previous section, Multi-Knowledge experiments represent process instances, and each experiment step represents an activity.

Experiment steps are defined by and conducted under the responsibility of a research team, coordinated by a Principal Investigator. Starting from a patients data sample, usually defined and collected in the first work phases, the experiment is set to conduct successive data analysis cycles, aimed at extracting new knowledge through the exploitation of full integration among heterogeneous data clinical, demographical, genomic and proteomic managed by a diverse set of researchers. Biomedical researchers and biostatisticians are the major members of the research team.

The data sample is populated through the execution of relevant data collection steps. As above mentioned, data collection steps are normally the first steps to be conducted. Data analysis steps form the core of the experiments analysis cycle. Through them, the data sample is successively analysed by different classes of researchers having different "scientific cultures" and backgrounds that use different analysis tools, work in different environments, at geo-

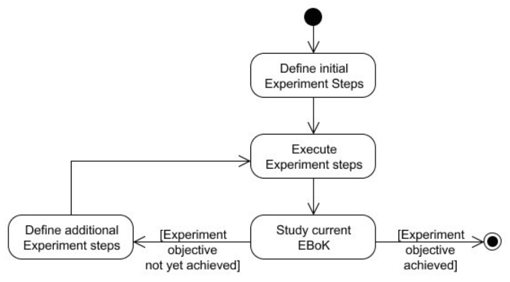
Define initial Experiment Steps
Execute Experiment steps
Define additional Experiment steps
[Experiment objective not yet achieved]
Study current EBoK
[Experiment objective achieved]

Figure 1. Multi-Knowledge workflow template.

graphically dispersed sites. Each of the data analysis step may generate new knowledge elements that contribute to create and successively expand an *experiment-related body of knowledge (EBoK)*. Based on an analysis of the EBoK (performed from their different scientific point of views) research team members can propose the execution of additional experiment steps or to further carry on the process.

Thus, the Multi-Knowledge workflow system:

- introduces experiment steps that are conducted by different researchers with diverse scientific background and cultures;
- supports the need of passing control back and forth from different researchers to perform data analysis steps relating to completely different mathematical foundations;
- allows the experiments consist of dynamical cycles of data collection and analysis that aim at progressively achieving the scientific goal initially stated for the experiment;
- concerns the collaboration among different teams, which are independently performing experiments in related areas.

The general template for Multi-Knowledge workflows complies with the activity diagram in figure 1.

When a team member, possibly after receiving a suggestion sent by another team member or by the principal investigator, decides to execute an experiment step, he/she:

- revises the proposed experiment step definition and possibly improves it based on her/his specific knowledge;
- executes the experiment step;
- adds an annotation, presenting the motivations for the experiment step as well as comments on steps execution and outcome.

The workflow engine reacts by logging the experiment step that has been executed, in terms of task identifier, ask parameters and used data set, and by recording the annotations produced by the team member that executed the Step.

Moreover, team members are allowed to browse the current experiment status, consisting of all the experiment steps conducted so far and related information. The system offers a comprehensive view of this knowledge, allowing to choose and extract graphic representations of different (including intermediate) steps of the experiment, to define and print reports based on the experiment status or on specific parts of it, and to perform statistics on logging information coming from different experiments managed within the system, in order to extract performance measures and identify best practices.

To capture such collaborative processes, involving multiple services within and across organizational boundaries, we consider the choreography approach to be the most effective. A choreography model describes, from a global point of view, the interactions between a collection of services in order to achieve a common goal, which is, in the context of Multi-Knowledge, the execution of data analysis pipelines that are used for comparing observed and predicted data, and include a wide variety of components for performing specific functionalities (*e.g.* querying databases, data transformations, data mining, simulations, etc.). Choreography may facilitate the need of the scientists to plug-in almost any scientific data resource and computational service into the workflow, inspect and visualise the data on the fly as they are computed, make parameter changes when necessary and re-run only the affected downstream components and capture sufficient metadata in the final products.

The Multi-Knowledge workflow engine (hosted by few selected nodes, managed by principal investigators) produces workflow descriptions, each one appearing as a composition of task descriptions. These tasks can be assigned to other available nodes (providing the required services) for immediate execution, but also their descriptions can be published in the network of participating nodes for deferred execution. In both cases a peer-to-peer infrastructure is appropriate, as it provides scalable mechanisms for service publication and discovery. The following section is devoted to the description of the solution we adopted for this aspect of the Multi-Knowledge project.

5 Peer-to-Peer Infrastructure

The Multi-Knowledge service-oriented architecture (MK-SOA) is based on Web Services, each of them having an observable behaviour which is described by WSDL. The ordering rules of the collaborative observable behaviours are specified by means of W3C's Web Services Choreography Description Language (WS-CDL), the ongoing standardization proposal in the area of service choreography.

The distributed middleware infrastructure is based on the peer-to-peer paradigm, since each Multi-Knowledge node must be autonomous and able to

- publish its data-oriented services (entry, integration, analysis);
- exchange messages with other nodes in order to find suitable services, representing experiment steps (activities), and request their execution, in the context of a complex experiment (process instance).

Each experiment is described by a WS-CDL document detailing roles, relationships, participants (implementing one or more roles) and activities. The node responsible for the definition of an experiment searches for nodes able to provide the service required by the first task. If several providers are available, the most reputed among them is chosen. Each peer executing a task has the responsibility of finding a node (or nodes) providing the service(s) required by the following step(s) in the experiment, and to send to that node(s) the workflow description document along with its partial results. All peers involved in the experiment send logging information to the peer which generated the experiment description (as described in section 4).

The adopted technology is JXTA, Sun Microsystem's set of open, generalized peer-to-peer protocols that allow a vast class of networked devices (smartphones, PDAs, PCs and servers) to communicate and collaborate seamlessly in a highly decentralized fashion. JXTA aims at reducing the complexity otherwise required to build and deploy peer-to-peer applications and services by providing an open source software platform and deployed virtual network.

In the context of JXTA project, the Department of Information Engineering of the University of Parma is responsible for the JXTA-SOAP component [6], currently allowing to to deploy, publish, discover and invoke Web Services in a network of JXTA peers.

In order to deploy its services, a peer has to instantiate and configure the related SoapServices (one for each hosted service), and to advertise the service interfaces in the network. The class diagram in figure 2 illustrates the classes and the relationships among them which are involved in this tasks. In particular, SOAPService is the class which must be associated to every service implementation, providing a common initialization method. Each service implementation contains a ServiceDescriptor, providing basic information which uniquely identify the service, and implements the ServiceLifecycle interface, which is used to pass a *Context Object* [11] (constructed by the user application) to the Web Service class. A security Policy is associated to each service implementation.

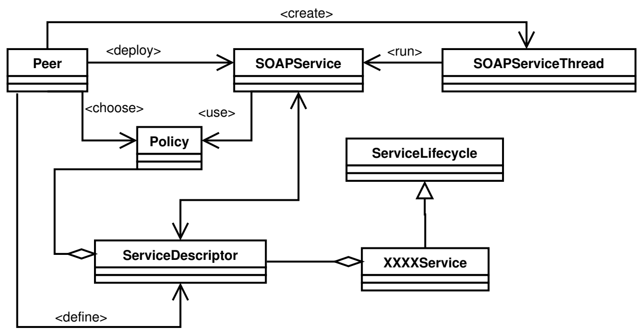

Figure 2. Class diagram for service deployment.

In JXTA-SOAP, Web Service interfaces are encapsulated in particular JXTA documents, the so-called ModuleSpecAdvertisements, which can be spread into the network using many different routing policies. Once the ModuleSpecAdvertisement has been filled with the service WSDL and the secure pipe tag, a context object is created and passed to the SOAPService instance. Moreoever, the peer spawns a SOAPServiceThread which waits for other peers' connections to the deployed service, on the public pipe associated to the ModuleSpecAdvertisement.

Service publication is a distributed process, which uses network nodes as a distributed repository (on the contrary, traditional UDDI registries are centralized interface description repositories). As for publication, service lookup is a distributed process, which can be conceptualized as message exchange between low-level JXTA modules.

JXTA peers are connected in such a way to form an overlay network in which peers with higher bandwidth and process capacity act as supernodes, assuming the responsibility of propagating messages, while peers with less capacities (leaf nodes) connect to supernodes and send them publication/query messages, but do not contribute to the overall routing process. Each leaf node is connected to a (single) supernode. On the other side, supernodes maintain many leaf connections, as well as a small number of connections to other supernodes. This hierarchy leads to scalable systems, in which supernodes shield leaf nodes from virtually all ping and query traffic.

The high degree of decentralization which characterizes the peer-to-peer approach introduces several security issues. Currently, JXTA-SOAP supports secure service invocation based on two ortogonal mechanisms. The first one, *transport-level security*, allows to create a TLS-based secure channel which guarantees the integrity and confidentiality of exchanged information, by means of mutual authentication between parties (using X509 certificates), and data encoding. The other approach is WSS-based *message-level security*, for which SOAP messages sent by service consumers contain security parameters (tokens) which are extracted by service providers to check for consumers' compliance with the security policy of the invoked service.

6. Application Example

To verify the correctness of the implementation, we spread a set of MK peers into the public JXTA network. MK peers were hosted by quite heterogenous machines, ranging from Intel Pentium IV and Xeon to AMD Athlon, with 512MB RAM or more, and running either Windows XP or Linux (kernel 2.6.x).

An example of MK network is represented in figure 3. All MK peers have the same capabilities, provided by JXTA plus JXTA-SOAP modules, but are allowed to deploy different services. The DataCollectionService (DCS) provides data storage and retrieval functionalities for local or remote users:

- physicians uploading raw clinical data;
- medical researchers downloading raw data and uploading integrated (clinical + microarray + ..) data;
- biostaticians downloading integrated data to perform some kind of analysis whose results are stored by means of a DCS, too.

The DataIntegrationService (DIS) allows to semantically integrate heterogenous data acquired from different sources (exposed by means of DCSs). Both DIS and DCS are implemented in Java and rely on open source technologies.

The DataAnalysisService (DAS) wraps the GenePattern [16] engine, providing operations for single task execution or pipelining. The DAS accepts a list of task description (including parameters) to be sequentially executed, and input data for the first task. Each requestor is provided with a VisualizationService (VS), based on GraphViz [9], which can be invoked by the DAS in order to show the graphical results of the data analysis process.

Figure 3 illustrates the publication process of DCS and DAS, initiated by peers belonging to different research groups. A third peer searches for both services, and finds the information about their location in the supernode network. Secure SOAP message exchange for service request/response is then perfomed by the involved peers (requestor and service provider).

In the final prototype some peers will be provided with the Workflow Engine, allowing to design and execute complex process instances composed by local and/or remote services, retrieved in the network.

The functional evaluation demonstrated the effectiveness of JXTA's distributed discovery service, and the responsiveness of JXTA-SOAP (which can be compared to the

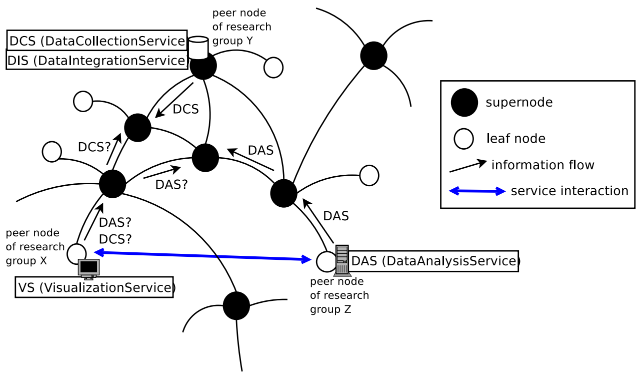
peer node of research group Y
DCS (DataCollectionService)
DIS (DataIntegrationService)
DCS
DCS?
DAS
DAS?
DAS
DAS? DCS?
peer node of research group X
VS (VisualizationService)
DAS (DataAnalysisService)
peer node of research group Z
supernode
leaf node
information flow
service interaction

Figure 3. Example of Multi-Knowledge peer-to-peer collaborative environment. DCS and DAS are published by peers which belong to different research groups (Y and Z). Another peer, belonging to group X, searches for DCS and DAS. The final interaction is peer-to-peer between DAS and VS.

most used Web Service engines, such as Tomcat+Axis and IIS+ASP.NET). An important aspect which has been proven is the ability of each leaf peer to autonomously find a new supernode, when the current one becomes overloaded or unreacheable. Thus the overlay network is always connected, and the performance of the message routing process depends only on the topology of the overlay network itself. This topic, with reference to JXTA's topologies and routing algorithm, has been discussed in our previous publication [7].

7. Conclusions and Future Work

In this paper we illustrated the collaborative environment proposed by the Multi-Knowledge project. We illustrated the generic template of Multi-Knowledge workflow processes, and motivated the adoption of the choreography model which implies a peer-to-peer infrastructure for connectivity and communication among service providers. Thus we described JXTA-SOAP, focusing on the implementation of its internal mechanisms, which are strictly related to communication and connectivity protocols provided by JXTA.

Next step is to implement the Workflow Engine on top of JXTA-SOAP. To this purpose, a promising tool is Pi4SOA [5], which provides WS-CDL support.

8. Acknowledgements

This work has been supported by the Multi-Knowledge project, which is funded by the European Commission in the context of the Sixth Framework Programme for Research and Technological Development (Project #027106, thematic area Information Society Technologies).

References

[1] ARTEMIS EU Project homepage. http://www.srdc.metu.edu.tr/webpage/projects/artemis/index.html.

[2] The Biological Web Services page. http://taverna.sourceforge.net/index.php?doc=services.html.

[3] COCOON EU Project homepage. http://www.cocoon-health.com.

[4] Multi-Knowledge EU Project homepage. http://www.multiknowledge.eu.

[5] Pi4SOA Project homepage. http://www.pi4tech.com.

[6] M. Amoretti, M. Bisi, and A. Panisson. JXTA-SOAP Project. http://soap.jxta.org.

[7] M. Amoretti, F. Zanichelli, and G. Conte. Performance evaluation of advanced routing algorithms for unstructured peer-to-peer networks. In *First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2006)*, October 2006.

[8] F. Banaei-Kashani, C. Chen, and C. Shahabi. WSPDS Web Services Peer-to-peer Discovery Service. In *The 2004 International Symposium on Web Services and Applications*.

[9] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software — Practice and Experience*, 30(11):1203–1233, 2000.

[10] A. Harrison and I. Taylor. Dynamic Web Services Deployment Using WSPeer. In *Proc. of the 13th Mardi Gras Conference*.

[11] A. Krishna, D. C. Schmidt, and M. Stal. Context Object: A design pattern for efficient middleware request processing. In *12th Pattern Language of Programming Conference*, September 2005.

[12] S. Krishnan, K. Baldridge, J. Greenberg, B. Stearn, and K. Bhatia. An end-to-end web services-based infrastructure for biomedical applications. In *6th IEEE/ACM International Workshop on Grid Computing*, November 2005.

[13] T. Liefeld, M. Reich, J. Gould, P. Zhang, P. Tamayo, and J. Mesirov. GeneCruiser: a Web Service for the annotation of microarray data. *Bioinformatics*, 18(21):3681–3682, 2005.

[14] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic Matching of Web Services Capabilities. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 333–347, 2002.

[15] I. Pyarali, M. Spivak, R. Cytron, and D. C. Schmidt. Evaluating and optimizing thread pool strategies for Real-Time CORBA. In *ACM SIGPLAN Workshop on Optimization of Middleware and Distributed Systems (OM 2001)*, June 2001.

[16] M. Reich, T. Liefeld, J. Gould, J. Lerner, P. Tamayo, and J. Mesirov. GenePattern 2.0. *Nature Genetics*, 38(5):500–501, May 2006.