数式処理 *J.JSSAC (2004)*
Vol. 10, No. 4, pp. 12 - 23

特集「大会発表論文」

On the Framework of Web-based Problem Solving Environments

K. Li, M. Sakai, Y. Morizane, M. Kono, M.-T. Noda*

Ehime University

(Received 2003/9/30 Revised 2004/1/24)

Abstract

The first design of *Lupin*, a layered framework of Web-based, common *Problem Solving Environments* (PSEs) is proposed and discussed. The idea of invoking Web technologies such as the emerging *Web services* for Lupin's approach; the conception of *mathematical Web services* and the implementation of Lupin based on the conception are briefly considered. A proof-of-concept system addressed by the combination of the current Web service protocols, MathML, Computer Algebra Systems (CASs), interactive math and the relevant XML technologies is also presented to check out the feasibility and disadvantages.

1 Introduction

The increasing diversity in scientific and engineering computation is strongly motivating the research of powerful *Problem Solving Environments* (PSEs) [3, 2, 17]. As complex software systems, the traditional PSEs are monolithic. The main objective of their designers is to provide a solution to a specific problem rather than identify common function and build a common PSE platform. This situation results largely a high-cost, heavy-coded ad hoc procedure that lacks of flexibility, portability and generality on PSE construction. Due to this, it is believed that the network-centric, interoperation-enabled common PSE mechanism that exploits the existing hardware and software resources needs to be considered. There are many notable works have been done to put forward PSE technologies on this way, most of them focus on the mathematical software integration that enables the interoperation over networks [1, 4, 5, 6]; while few of them aim at the sophisticated framework that supports the whole lifecycle of PSE construction. Although PSE technology is improving quickly, there are still lack of significant approaches that are, not only

*{likai,masato,morizane,kono,noda}@hpc.cs.ehime-u.ac.jp

really language-, vendor- and platform neutral that truly address the large-scaled, seamless integration; but also simple, easy-to-use, and widely supported by the whole academic community. Obviously, much work is still ahead before achieving a common fundamental network-centric, distributed mechanism for the easy-but-efficient PSE construction.

Internet/Web is changing the way of processing information; it is also expected to play the role of PSE platform and to facilitate the mechanism of PSEs creation. In recent years, XML and its compliant Web services [7] are enabling applications *discovery*, *registration*, and *invocation* over the Internet. Hence we predicate that it's the time to consider the mechanism of PSEs to be Web-based and truly support the "cheap and effective" computation by enabling the problem solver's portability, reusability and search-ability. Towards this end, *Lupin* is our ongoing research subject in Ehime University, Japan that aims at:

- Establishing a Web-based mechanism and framework that allows easy and systematic creation and construction of computational PSEs.
- Exploiting the standard Web technologies to support the infrastructure.
- Implementing a prototype to demonstrate the feasibility.

In this paper, a common framework of Web-based computational PSEs is briefly given. The layered approach enables the independent development and deployment of Internet accessible PSE components (we call them *Lupin services*) by the *service provider*; and actual PSE construction by the *PSE provider* on the base of *Lupin service composition*. Based on the idea and concept of *mathematical Web service*, which is extended from e-business-oriented Web service technology, a prototype is also implemented to demonstrate the feasibility of Lupin architecture. This proof-of-concept system is realized by the aggregation of Web service technologies such as SOAP [8], WSDL [9], UDDI [10], as well as the mathematical protocol MathML, Computer Algebra Systems, interactive math [16] and the relevant XML technology. Various discussions are carried out according to the implementation and the improved proposals are also considered.

2 Lupin's initial design and its conceptual architecture

2.1 Overview

As a Web-based approach, Lupin focus on the two key elements that a PSE contains: 1) the user interface and 2) packages of computing kernels. In the Web-centric environment, depending on target class of problems, the process of a PSE construction is essentially the process of choosing and locating the certain computation kernels and binding them in an appropriate flow. Based on this conception, following elements are considered to play the critical roles under Lupin's framework.

- the computing kernels, that are called *Lupin services*, are Internet accessible.
- a mechanism of Lupin services *discovery*, which allows those geographically distributed Lupin services to be correctly selected and located.
- a mechanism of Lupin service *binding* to enable the integration and interoperation.
- a mechanism of *PSE composition* due to a certain definition of selected Lupin services and the corresponding interface generation for the end user.

In Figure 1, a layered stack is given. It shows that the *Lupin service provider*, the *PSE provider*, and the *end user* are three key entities in Lupin's framework and sit at different layer respectively—that means all the operations such as Lupin service developing and maintaining, PSE creation and generation as well as the use of PSE can be performed independently, in every time, at everywhere on a global wide.

2.2 Architecture

Based on the stack, Lupin is designed to have a 3-block architecture to facilitate Web-based PSEs construction. They are called the *Lupin service generation mechanism*, the *Lupin discovery mechanism* and the *PSE composition mechanism* respectively. In a typical scenario, Lupin service generation mechanism is considered to supply the *service providers* various capabilities to develop and deploy the actual Lupin services that are Internet accessible; and the other two parts are expected to facilitate the *PSE providers* to easily create their PSEs by enabling the appropriate Lupin services discovery and integration, as well as the PSEs user interface generation.

- *Lupin service generation mechanism*: Developed for particular purposes, Lupin services are separately owned and located and are available to participate in other systems via different interfaces. Lupin service generation mechanism aims to achieve the easy development of Lupin service as a back-end, and its deployment to be Internet accessible. Some contributions can be listed to support the distributed approaches either in encoding protocol level [1, 5], and program interface level [6, 4]. As an evolving standard protocol, SOAP [8] is garnering a great deal of interest from industry to address the XML messaging-based distributed computation.
- *Lupin service discovery mechanism*: It aims to organize Lupin services into a coherent collection to enable *discovery.* This is regarded to be addressed by the recommender system [3]. In a typical scenario, it accepts the query information from the PSE provider via interfaced *service browser*, searches for the satisfied one according to the existing service description, and returns the searching result for actual PSE composition. In the

Figure 1: Lupin's stack and its conceptual architecture

Web environment, we believe that XML-based meta-data technologies can provide us informative approaches towards the mathematical service description and discovery.

- *PSE composition mechanism*: After gathering all necessary Lupin services, the next step should be how to organize them to work with each other interactively in an appropriate way. PSE composition mechanism is motivated to address this goal. In a general paradigm, the PSE builder obtains the selected Lupin services as the "raw materials" and organizes them according to the certain application, together with other technologies that support math on the web. Then, user interface (which can be a standalone or the regular Web browser compliant) will be defined and generated as the client site front-end.

3 Implementation

3.1 From Web service to mathematical service

Significant facilities should be considered to implement Lupin's conceptual architecture and to address the math on the web. Among many others, for example, *MathML* and *OpenMath* are two fast-growing protocols that deal with the viewing and exchanging of mathematical contents on the Web. Moreover, the list of MathML-/OpenMath-compliant software, such as *WebEQ* and *Mathplayer* [16] which can support interactive math operations, is also increasing fast. They are considered to play the role of mathematical data exchange in Lupin's framework.

Mathematical contents delivery does not sufficiently cover Lupin's distributed computing mechanism. Among the Web-based technologies, we focus on the term *Web service*, which is an emerging e-business framework that can interface a collection of operations that are network-accessible through XML messaging [7]. There are three main entities in Web service architecture: the *service provider*, who hosts the actual service that is accessible over the Web; the *service requestor*, an application that requires certain function to be satisfied; and the *service registry*, a searchable mechanism for service description and discovery. Some XML-based standards such as SOAP [8], WSDL [9], and UDDI [10] are being developed to address its conceptual view.

Web services technology extends the application of Web from pure html-based contents to programming language-, programming model-, and system software-neutral platform. Hence it strongly motivates us to expand its features from the e-business domain, to a more powerful approach of a new distributed computing mechanism, say, *mathematical Web service*, to support Lupin implementation: to exploit SOAP as the underpinning for Web-based distributed computation, together with other technologies that deal with *math on the Web*, to serve the PSE composition mechanism and generation mechanism; to adopt WSDL

Figure 2: Lupin's implementation based on Web service protocols

for Lupin service description; and consider to use UDDI to support the implementation of Lupin discovery mechanism. Figure 2 shows our first proposal to implement Lupin's architecture.

3.2 Proof-of-concept system

Based on the presented Lupin framework, a prototype has been built to check out the feasibility. We focus on three procedures that illustrate the whole lifecycle of Lupin application: (1) the process of Lupin service creation/deployment by SOAP infrastructure in the Lupin service generation mechanism; (2) service registration and discovery by UDDI registry in the Lupin discovery mechanism; and (3) the actual service binding and user interface by Lupin PSE runtime in the PSE composition mechanism.

The testbed is built in Java. The Apache Tomcat is adopted to active the Servlet-enabled web server. Among available SOAP implementations, we chose also an Apache package called AXIS [14]. Apache Tomcat Web server and AXIS constructed the backbone of our experimental environment. Figure 3 is the whole diagram of the testing system.

lupin_exep_big_ok.eps

Figure 3: Operation flow of Lupin application

3.2.1 Lupin service development, deployment and registration

In this system, all the Lupin services are considered to talk MathML as the default mathematical protocol. For example, Lupin service `exeAsirFactorService` is implemented by a Java class `FactorService`. It accepts polynomial as the input, invokes Risa/Asir as the computing engine to compute the factorization of the polynomial, and returns the answer with MathML expression. The format conversion between MathML and the native mathematical expression is addressed by our XSL library that can be processed by Java parser. With AXIS, it can be deployed online by using a `wsdd` (Web service deployment descriptor) file which specifies certain properties of the service:

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
            xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
<service name="exeAsirFactorService" provider="java:RPC">
  <parameter name="className" value="webservice.asir.FactorService"/>
  <parameter name="allowedMethods" value="exe"/>
</service>
</deployment>
```

After being deployed, the service can be located from the Web by the actual URI:
`http://localhost:8001/axis/services/exeAsirFactorService.`
The corresponding WSDL file `exeAsirFactorService.wsdl` can also be generated by the attached tool `Java2wsdl` provided by the AXIS SOAP infrastructure, or just could be simply retrieved by going to:
`http://localhost:8001/axis/services/exeAsirFactorService?wsdl`.
The WSDL file, which enables the dynamic interface generation to invoke the deployed Lupin service, plays a critical role in Lupin implementation.

3.2.2 Lupin service retrieval

To make Lupin service to be searchable, it should be firstly registered to Lupin's recommender system with corresponding WSDL and relevant data. The recommender system is here performed by a UDDI service registry. In our experiment, systinet's WASP UDDI registry [11] handles the operation to demonstrate that whether it makes sense working as the discovery mechanism of mathematical services. WASP UDDI supplies a Web interface that allows the access to the UDDI-specified registry. Both of service registration and searching operations are supported. The current UDDI specification can only support a very simply query to achieve the service searching, e.g., *service name*, *business name* and some default standard business *taxonomy*. There is still not default mathematical taxonomy system defined in UDDI that accommodates the mathematical application discovery. Hence in our experiment, a temporary mathematical taxonomy system based on GAMS [15] is added due to the extended functionality of WASP UDDI. The result shows that the successful discovery can be achieved by querying the registered service name *exeAsirFactorService*, as well as its corresponding classification defined in the mathematical taxonomy.

3.2.3 PSE composition

We are now developing the *Lupin PSE builder*, a Java toolkit that completely supports the Web service-based PSE composition. The detail discussion will appear in our related report. Here we only focus on the experiment, which is involved in two parts: the PSE interface generation and the Lupin service binding. The former one aims at the friendly user interface that made by the PSE provider to address the easy-to-use PSE frond-end; while the later one targets the programmed integration of the selected Lupin services.

The service invocation is carried out by *Lupin Service Binding API*, which is a part of Lupin PSE runtime and currently implemented in Java. Based on AXIS and DOM parser, it can dynamically generate the interface for SOAP remote call when given the URL of WSDL file of certain Lupin service, e.g., the `exeAsirFactorService.wsdl`. In this implementation, every Lupin service is a SOAP server and is responsible for processing the request message and formulating a response. The response message is received by the networking infrastructure on the service requestor's node and can be converted from XML message into certain object that fits the client.

We've also made an effort to show that the PSE provider could be supported by the existing *interactive math* approaches, to build the application-oriented PSE user interface. Achievements of many aspects of "math on the web" are providing the significant solutions. As an example, WebEQ [16] interactive math technology based on Java applets that enables MathML processing has been used. With it, the `exeAsirFactorService` can be invoked by a series of user-friendly operations without necessity to care about what kind of computing

engines are working, nor necessity to learn syntax rule of certain CASs. The SOAP-based messages transmitted during the distributed computation are shown as follows:
Request message:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <exe soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <cmml xsi:type="xsd:string">
    <math><apply><minus/><apply><power/><ci>x</ci><cn>6</cn></apply><apply><power/>
    <ci>y</ci><cn>6</cn></apply></apply></math>
   </cmml>
  </exe>
 </soapenv:Body>
</soapenv:Envelope>
```

Response message:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <exeResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <exeReturn xsi:type="xsd:string">
    <math><apply><times/><apply><times/><apply><times/><apply><times/><cn type="integer">
    1</cn><apply><plus/><apply><plus/><apply><power/><apply><ci>x</ci></apply><cn type=
    ...
    <apply><ci>y</ci></apply></apply></apply><apply><plus/><apply><ci>x</ci></apply><apply>
    <ci>y</ci></apply></apply></apply></math>
   ...
</soapenv:Envelope>
```

4 Remark

The proof-of-concept system shows that the process of Lupin service invocation can be sufficiently achieved by WSDL-based interface generation and SOAP-based data exchange. Hence it is believed that the Web service standard protocols and the relevant XML-based technology can significantly benefit Lupin's Web-based PSE mechanism by enabling software reusability, portability and inter-operation.

Lupin's critical concept is to enable the independent development, deployment, discovery and invocation of Internet accessible Lupin services. When well implemented, Lupin will empower the PSE providers by eliminating many technical difficulties and exploiting the existing computing resources on their PSE construction. From this perspective, as a common PSE framework addressed by XML artifacts, Lupin can be applied by a wide range of field. For example, e-learning can benefit from Lupin's infrastructure. The topic of *author-once-learn-anywhere* (AOLA) for the teacher, researcher, and students in mathematical community has been discussed for a long time, but yet, it is not clear how AOLA

Figure 4: Interface generation and input/output from Lupin service

can be achieved. Lupin is expected to facilitate the answer: educators can produce education contents by invoking/assembling separately located Lupin services—answer checking, graph plotting, exercise generation etc., without necessary to learn about the actual programming and coding for the invocation. Those Lupin service, on the other hand, are created and published by service providers and are accessible due to the standard XML-based protocol. Our later research will focus on this application.

More over, due to our works, we claim that the UDDI registry does not completely meet our needs of Lupin service discovery, because of UDDI's poor description facility on mathematical issues. Thus, a more informative and semantic approach that efficiently enables the mathematical service discovery is needed. We are now developing a new Lupin service registry based on the emerging Mathematical Service Description Language (MSDL) [12] and the relevant semantic Web technologies [13] to urge our work.

5 Conclusions and future works

In this paper, we have discussed the first design of Lupin, which aims to build the common, significant computational PSE framework based on the Web. Driven by XML artifacts, Lupin highlights its characteristics to be open, flexible, portable and robust for wide-range of PSE construction, and will empower the PSE providers by eliminating many technical difficulties and exploiting the existing computing resources. Conform to the design, an implementation based on Web service protocols (SOAP, WSDL, UDDI, etc.) and relevant XML compliant technologies to address Lupin's whole architecture is also considered, and a proof-of-concept system is proposed to check out the feasibility.

Lupin is evolving, our immediate work is to complete the implementation of Lupin service registry and its relevant service browser to facilitate current Lupin discovery mechanism. The development of the Lupin PSE builder, a toolkit pack consists of Java-based LSB (Lupin Service Binding) API as well as the Lupin service flowing markup and its generator is also on-going to accommodate the PSE composition mechanism. Currently targeting at Web-based distributed computation and interactive mathematic education, Lupin will grow up together with the progress of Internet technology.

References

[1] Wang, P. S.: Design and Protocol for Internet Accessible Mathematical Computation. In *Proc. ISSAC'99*, ACM Press, pp. 291–298, 1999.

[2] Lakshman, Y. N. et al.: Software Components using Symbolic Computation for Problem Solving Environments. In *Proc. ISSAC'98*, ACM press, pp. 46–53, 1998.

[3] Houstis, E., and Rice, J. R.: On the Future of Problem Solving Environments, http://www.cs.purdue.edu/people/jrr, 2000.

[4] Liao, W., Lin, D., and Wang, P. S.: OMEI: An Open Mathematical Engine Interface. In *Proc. ASCM'01*, World Scientific Press, pp. 82–91, 2001.

[5] Maekawa, M., Noro et al.: The Design and Implementation of OpenXM-RFC 100 and 101. In *Proc. ASCM'01*, World Scientific Press, pp. 102–111, 2001.

[6] JavaMath, `http://javamath.sourceforge.net`

[7] Kreger, H.: Web Service Conceptual Architecture (WSCA 1.0). IBM Software Group, `http://www-3.ibm.com/software/solutions/webservices`, 2001.

[8] Seely, S.: SOAP: Cross Platform Web Service Development Using XML. Prentice Hall.

[9] WSDL (Web Services Description Language), `http://www.w3.org/TR/wsdl`

[10] McKee, M., Ehnebuske, D., and Rogers, D.: UDDI Open Draft Specification. `http://www.uddi.org`, 2001.

[11] Systinet WASP, `http://www.systinet.com`

[12] MSDL (Mathematical Service Description Language), `http://monet.nag.co.uk`

[13] Montebello, M., and Abela, C.: DAML enabled Web Service and Agents in the Semantic Web. `http://www.daml.org`

[14] AXIS (Apache eXtensible Interaction System), `http://ws.apache.org/axis`

[15] GAMS mathematical taxonomy, `http://gams.nist.gov/Taxonomy.html`

[16] WebEQ, `http://www.dessci.com`

[17] Li, K., Zhi, L. H., and Noda, M.-T.: On the Construction of a PSE for GCD Computation. In *Proc. ASCM'01*, World Scientific Press, pp. 76–81, 2001.