工学ジャーナル 第4巻
2005年3月
指定投稿論文

Lupin: Towards the Framework of Web-based Problem Solving Environments[1]

Kai LI[2], Masato SAKAI[2], Yukihiro MORIZANE[2], Masahiro KONO[2] and Matu-Tarow NODA[3]

The research of powerful *Problem Solving Environments* (PSEs) is strongly motivated by the increasing diversity in scientific and engineering computation. In this paper, the first design of *Lupin*, a layered framework of PSEs construction based on the Web is proposed and discussed. The idea of invoking Web technologies such as *XML* and 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.

Key words: Problem Solving Environments, Web services, XML, Computer Algebra System

1. Introduction

The research of powerful *Problem Solving Environments* (PSEs) is strongly motivated by the increasing diversity in scientific and engineering computation [3, 2, 19]. As complex software systems, the traditional PSEs are monolithic. The main purpose 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. Among those efforts, most of their focuses are on the mathematical software integration that enables the interoperation over networks [1, 4, 6, 5], 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 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. On the other hand, PSE's other critical facilities such as the semantic problem definition, automatic or semi-automatic problem solver discovery are also of great challenges and have not been significantly addressed yet. Obviously, much work is still ahead before achieving a common fundamental network-centric 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

[1]The best paper award of ATCM'2003 (Proceedings of ATCM'2003, pp.276–285, 2003)
[2]Graduate School of Engineering, Ehime University
[3]Department of Computer Science, Ehime University

原稿受理 平成16年11月1日

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, we briefly present a common framework of Web-based computational PSEs. A layered approach is given to enable 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*. A prototype has been implemented to demonstrate the feasibility of Lupin architecture. This proof-of-concept system is supported by the combination of Web service technologies such as SOAP [10], WSDL [11], UDDI [12], as well as the mathematical protocol MathML [9], Computer Algebra Systems (CASs) [6], interactive math [18] and the relevant XML technology. Various discussions are carried out according to the implementation and the improved proposals are considered for future research.

2. Lupin overview

2.1 basic idea

The initial thoughts of Lupin focus on the two key elements that a PSE contains: 1) the user interface and 2) packages of computing kernels - the computing engines, plotting applications, databases and other programs. In the Web-centric environment, it is expected that one kernel can contribute to a number of user interfaces that designed for different purpose and different end users - computer modeling for physical phenomena, numerical simulation, engineering computation, mathematical education and so on. In another word, 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. Thus, via a carefully designed and implemented user interface created by the PSE provider, the end users can get their desired computation to be well defined and solved transparently over the Web; different user interfaces and its bound computing kernels provide different PSEs for different class of computations. Based on this conception, the 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 actual integration and interoperation.

Lupin service consumption	End user layer
- PSE interface generation. - Lupin service flowing management. - Lupin service binding. - Lupin service discovery.	PSE provider layer
- Lupin service deployment. - Lupin service creation.	Service provider layer

Figure 1: Lupin's stack

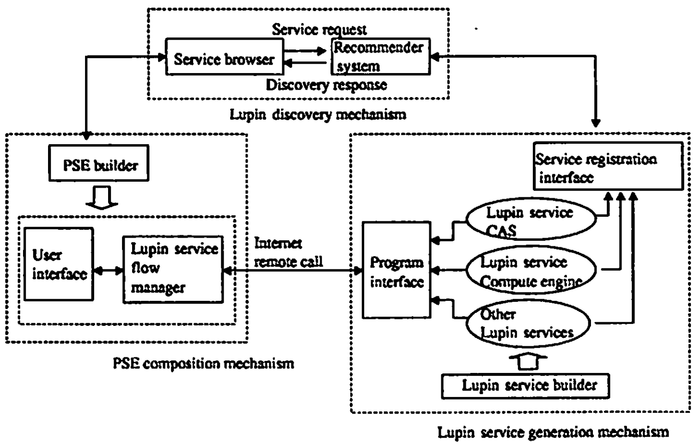

Figure 2: Lupin's conceptual architecture

- a mechanism of *PSE composition* due to a certain definition of selected Lupin services and the corresponding interface generation for the end user.

Figure 1 shows the conceptual stack to illustrate the layered relation of the above elements. It figures out 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 over the Internet.

2.2 Lupin framework and its usage diagram

Based on the above conception, Lupin is designed to have the following architecture to facilitate Web-based PSEs construction (Figure 2). It consists of 3 main blocks, which 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.

2.2.1 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, 6], and program interface level [5, 4]. As an evolving standard protocol on the Web, SOAP [10] is garnering a great deal of interest from industry to address the XML messaging-based distributed computation.

2.2.2 Lupin service discovery mechanism

Lupin service discovery mechanism aims to organize Lupin services into a coherent collection to enable *discovery*. It should be based on the semantic match between a declarative description of the Lupin service being sought, and a description of the Lupin service being offered. In Lupin, the discovery mechanism is regarded to be addressed by the recommender system [3]. Generally, the recommender system contains an agent accessible registry, which holds Lupin service descriptions registered by service provider. 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 together with the binding information to the PSE provider for PSE composition. In the Web environment, we believe that XML-based meta-data technologies can provide us informative approaches towards the mathematical service description and discovery.

2.2.3 PSE composition mechanism

In the conceptual architecture of Lupin, the PSE provider contacts the recommender system via Lupin service browser to seek any Lupin services that contribute to the certain computation. 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 typical scenario, 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.

We note that, from the perspective of PSE, it is not sufficient to have the only three parts to define a significant PSE infrastructure. Our objective is to extract the most critical 3 entities from the framework of PSE to establish the backbone, and to demonstrate whether it makes sense to shift PSE conception to the Web environment, using the XML-based technology to support the mathematical PSE construction.

3. Implementation

3.1 Available technologies

We focus on the following available approaches that can potentially benefit Lupin's architecture.

The topic about *math on the web* has been well researched and developed in recent years. Among many others, for example, *MathML* [9] and *OpenMath* [8] are two fast-growing protocols that address the viewing of mathematical contents on the Web, as well as allowing the rich math object to be exchanged between programs. Moreover, the list of MathML-/OpenMath-compliant software is also increasing fast, such as *WebEQ* and *Mathplayer* [18] can support interactive math operations based on MathML; and series of Phrasebooks implement the conversion

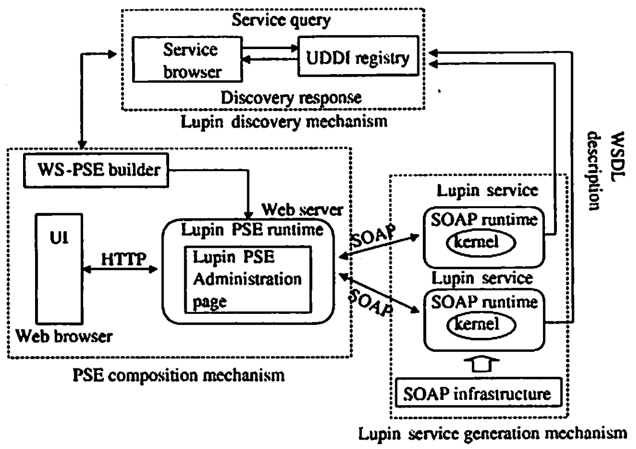

Figure 3: Lupin's implementation based on Web service protocols

between OpenMath objects and software applications.

The term *Web service* is an emerging e-business framework that can interface a collection of operations that are network-accessible through XML messaging [7]. There are 3 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; where service providers publish their service descriptions to, and service requestors find service and obtain binding information from. Some XML-based standards such as Simple Object Access Protocol (SOAP) [10], Web Service Description Language (WSDL) [11], and Universal Description, Discovery and Integration (UDDI) [12] are being developed to address its conceptual view.

3.2 Lupin implementation based on Web service protocols

Widely supported by the industry community, the Web services technology extends the application of Web from pure html-based contents to Programming language-, programming model-, and system software-neutral platform. Hence we are strongly motivated to expand its characteristics from the e-business domain, to a more powerful approach of a new distributed computing mechanism, say, *mathematical Web service*, to support Lupin implementation. Based on this thought, our first attempt is to implement Lupin's framework with series of standard Web service protocols. That is, to exploit SOAP as the most fundamental 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 as the standard protocol for Lupin service description; and consider to use UDDI to support the implementation of Lupin discovery mechanism. Our later technical reports will focus on the detailed discussion for each Lupin mechanism respectively. Figure 3 shows our first proposal to implement Lupin's architecture with Web service standards.

4. Proof-of-concept system

A testing prototype has been built on the base of Lupin framework. Through a simple example, highlighted works are listed to 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

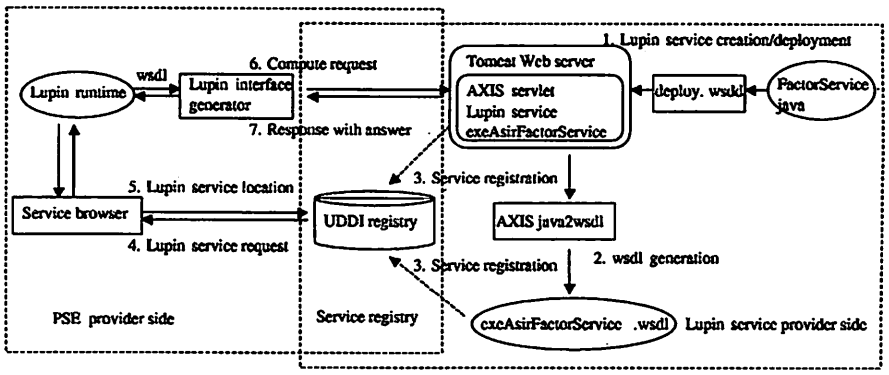

Figure 4: Operation flow of Lupin application

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 experimental testbed has been built in Java. On the consideration of Web server, the Apache Tomcat is adopted. Tomcat is a highly configurable server that supports Java servlets, which can well handle the Lupin service that one wants to make accessible. Among those available approaches that enable SOAP application, we chose also an Apache package called AXIS [16]. Apache Tomcat Web server and AXIS constructed the backbone of our experimental environment. Figure 4 is the whole diagram of the testing system.

4.1 Service development, deployment and registration

All the Lupin services in the system are considered to talk MathML as the default mathematical protocol. As a simple example, Lupin service `exeAsirFactorService` is implemented by a Java class `FactorService`. It accepts MathML-based polynomial as the input, invokes Risa/Asir [6], a local Computer Algebra System, to compute the factorization of the polynomial, and returns the computing 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 by AXIS, the service can be located from the Web by the actual endpoint. In our case, the service is available at:

`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. To make the WSDL file to be searchable, it should be firstly registered to Lupin's recommender system, which is here performed by a UDDI service registry. In our experiment, the operation is handled via systinet's WASP UDDI registry [13].

4.2 Service discovery

Under Lupin's architecture, The operation of service-searching is done by PSE provider through the recommender system that enables mathematical Web service discovery. Considering the objective of UDDI, our testing system adopted the "pure UDDI registry" provided by systinet's WASP UDDI [13] 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 a mathematical taxonomy needs to be added. In our experiment, a temporary mathematical taxonomy system based on GAMS [17] 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.

4.3 Service binding and PSE composition

Another important issue of our experiment is how to achieve the actual Lupin service binding when we obtained the WSDL file from UDDI registry; and how to construct the PSE by generating the user interface binding with the distributed Lupin services. We are now developing the *Lupin PSE builder*, a Java toolkit in Lupin's framework that completely support 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 2 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 operation of 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' 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 [18] 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>8</cn></apply><apply><power/>
    <ci>y</ci><cn>8</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><power/><apply><ci>x</ci></apply><cn type="integer">2</cn>
    </apply><apply><power/><apply><ci>y</ci></apply><cn type="integer">2</cn></apply>
    </apply></apply><apply><plus/><apply><power/><apply><ci>x</ci></apply><cn type=
    "integer">4</cn></apply><apply><power/><apply><ci>y</ci></apply><cn type="integer">
    4</cn></apply></apply></apply><apply><minus/><apply><ci>x</ci></apply><apply><ci>y
    </ci></apply></apply></apply><apply><plus/><apply><ci>x</ci></apply><apply><ci>y</ci>
    </apply></apply></apply></math>
   </exeReturn>
  </exeResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

5. Conclusions and Future works

We have discussed the design and architecture of Lupin, which aims to build the significant Web-based PSE framework in this paper. Enabling the independent development, deployment, discovery and invocation of Internet accessible PSE components (Lupin services), Lupin will empower the PSE providers by eliminating many technical difficulties and exploiting the existing computing resources on their PSE construction. According to the design,

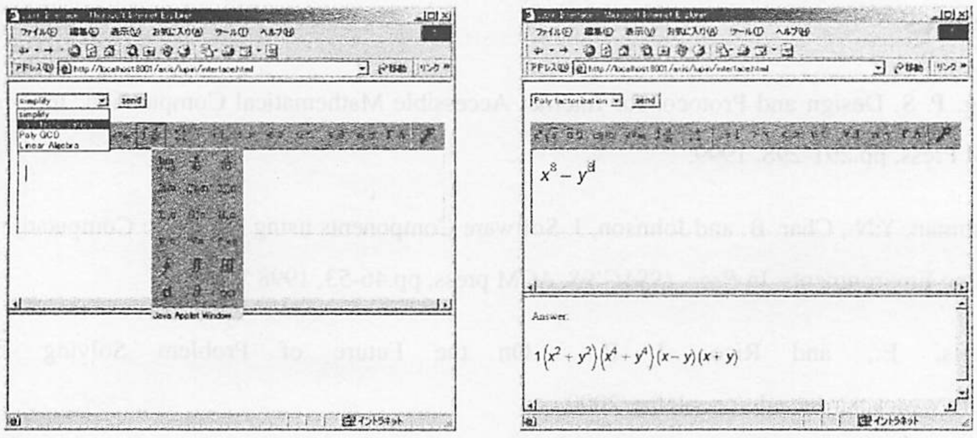

Figure 5: Interface generation and input/output from Lupin service.

an implementation based on Web service protocols (SOAP, WSDL, UDDI, etc.) and relevant XML compliant technologies has been considered to address the whole architecture, and a prototype has also been proposed to check out the feasibility.

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. However, the UDDI registry does not completely meet our needs of Lupin service discovery, because of UDDI's poor description facility on mathematical issues. From our perspective, the process of discovery should be carried out by certain service descriptions: one is the service declaration that offered by the service provider, while another is the query that a PSE provider seeks for. Current specification of UDDI includes very little information to describe service, limiting itself to the service name, the business name and some standard taxonomy. Even though UDDI allows services to refer to a set of attributes, called "Tmodel" (for example, in our experiment, we referred Tmodel to the Lupin service's WSDL file), but still lacks mechanism to support a flexible search. Due to this, a more informative, efficient and semantic approach which enables the mathematical service discovery is needed. We are now concentrating on a new Lupin service registry that is based on the emerging Mathematical Service Description Language (MSDL) [14] and the relevant semantic Web technologies [15], to extend our work. We predicate that, based on the mathematical service-oriented description and the semantic ontology markup approaches, Lupin discovery mechanism can be promoted to address a more effective and dynamic service discovery.

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., Char, B. and Johnson, J. 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] JavaMath, `http://javamath.sourceforge.net`

[6] OpenXM(Open message eXchange protocol for Mathematics), `http://www.openxm.org`

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

[8] OpenMath, `http://www.openmath.org`

[9] Mathematical Markup Language, `http://www.w3.org/Math`

[10] SOAP(Simple Object Access Protocol), `http://www.w3.org/TR/soap`

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

[12] UDDI(Universal Description, Discovery and Integration), `http://www.uddi.org`

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

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

[15] DAML, `http://www.daml.org`

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

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

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

[19] 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.