| | |
|---|---|
| Title | Lupin : from Web Services to Web-based Problem Solving Environments (Computer Algebra : Algorithms, Implementations and Applications) |
| Author(s) | Li, Kai; Sakai, Masato; Morizane, Yukihiro; Kono, Masahiro; Noda, Matu-Tarow |
| Citation | 数理解析研究所講究録 (2003), 1335: 149-156 |
| Issue Date | 2003-07 |
| URL | http://hdl.handle.net/2433/43351 |
| Right | |
| Type | Departmental Bulletin Paper |
| Textversion | publisher |

数理解析研究所講究録 1335 巻 2003 年 149-156



# Lupin: from Web Services to Web-based Problem Solving Environments

K. Li, M. Sakai, Y. Morizane, M. Kono, and M.-T.Noda
Dept. of Computer Science, Ehime University*

**Abstract**

The research of powerful *Problem Solving Environments* (PSEs) is strongly motivated by the increasing complexity in scientific and engineering computation. In this paper, an initial design of *Lupin*, a framework of PSEs construction based on Web technologies is proposed and discussed. We predicate that common infrastructure of PSEs can be powerfully supported by the Web, which is moving from being a collection of static *pages* toward a collection of dynamic *services* that interoperate through the Internet. The idea of invoking Web technologies such as *XML* and the emerging *Web services* for Lupin's approach is briefly discussed; A proof-of-concept system based on the current Web service protocols is also presented to check out the term *mathematical Web service* and how can Lupin benefit from it. Related issues are discussed based on the experiments and new proposals are offered for future research.

## 1 Introduction

The research of powerful *Problem Solving Environments* (PSEs) is strongly motivated by the increasing complexity in scientific and engineering computation [3, 2, 14]. 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. Hence the construction of PSEs still remains largely a high-cost, heavy-coded ad hoc procedure that lack of either flexibility, portability and generality. Due to this, it is believed that the network-centric, distributed approaches that exploit the existing hardware and software resources will be an attractive direction of further PSEs development. There are many notable works have been done to put forward PSE technologies on this way, most of their focus are on the problem solver integration that enables the interoperation over networks [1, 4, 6, 5]. However, the potential problem might be that those interactive, distributed approaches will not be well performed until being international standards and to be widely supported by the whole academic community. On the other hand, except the integration technologies, other critical PSE facilities such as the semantic problem definition, automatic or semi-automatic problem solver location and selection have not been significantly addressed yet. One of the reasons is also that the distributed components are language-, vendor- and platform neutral and hard to be communicated with and selected in a simple, standard way. Obviously, a common fundamental mechanism that enables the network-centric computing that built on wide-supported standards is needed.

Internet is changing the way of gathering information; it is also expected to be able to upgrade the mechanism of PSEs construction. Emerging Web technologies such as Web services [7] that based on XML are enabling the capabilities of 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 that truly supports the "cheap and effective" computation by enabling the problem solver portability, reusability and search-ability. Towards this end, Lupin is an 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.

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

- Exploiting the standard Web technologies to support the infrastructure.
- Implementing a prototype to demonstrate the feasibility.

In this paper, we briefly present an initial common architecture of constructing Web-based computational PSEs under Lupin's framework. 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 following process of *Lupin service composition*: (1)to search for the services that Internet accessible and that can be used to contribute the solution; (2)to locate the corresponding services that with the information of binding; and (3) the generation of the interface to invoke the service to get it work for the user. A testing prototype has been implemented as a proof-of-concept system to demonstrate the feasibility of Lupin architecture that supported by Web service technologies such as SOAP [8], WSDL [9], UDDI [10]. Various discussions are done according to the implementation and the new proposals are offered for future research.

## 2 Lupin overview

The basic idea of Lupin is due to the following thought: generally, as a integrated or interactive computing system, there are two key elements that a PSE contains: the user interface and 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 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, without necessary to conc ern about such questions as: where is the data needed for this computation? Which computer is used? Where is the computing engine located and so on. Different user interface and its bound computing kernels provides different PSE for different class of computations. 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 actual integration and interoperation.
- a mechanism of *PSE interface generation* due to a certain flow definition of selected Lupin services for the end user.

To perform the above operations in an interoperable manner, a conceptual stack is shown to illustrate the layered relation. The upper layer build upon the capabilities provided by the lower layers. Within the conceptual stack, it is noted that the *Lupin service provider*, the *PSE provider*, and the *end user* is sitting at different layer respectively - that means all the operations of Lupin service developing and maintaining, PSE creation and generation, and the use of PSE can be performed independently and interoperate one another on a global wide.

## 3 Lupin architecture and its usage diagram

In order to achieve the above conception of Web-based PSEs construction, Lupin is designed to have the following architecture. There are 3 main parts, which is called the *Lupin service generation mechanism*, the *Lupin discovery mechanism* and the *PSE composition mechanism* that plays the key role respectively.

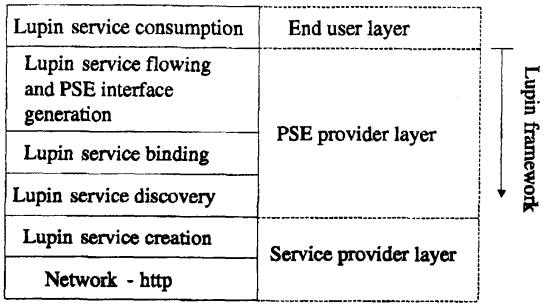


Figure 1: Lupin's stack

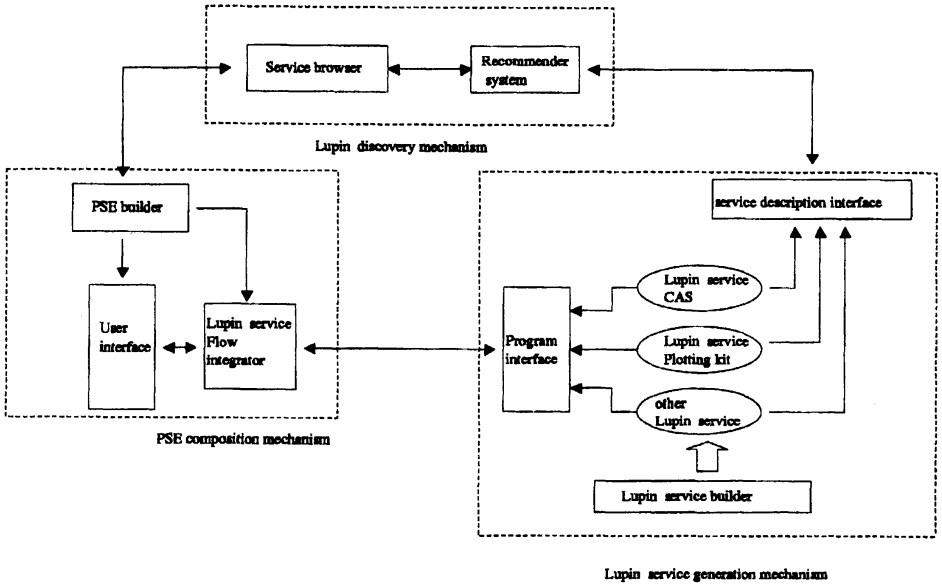


Figure 2: Lupin's conceptual architecture

In a typical scenario, Lupin service generation mechanism is used by the service provider to develop the actual Lupin service that is Internet accessible; and the other two parts are used by the PSE provider to achieve the appropriate Lupin service selection and location and the PSE user interface generation.

## 3.1 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 facilitate the easy development of Lupin service as a back-end and its deployment to be Internet accessible. There are dozens of contributions can be listed to support the distributed mechanism either in encoding protocol level [1, 6], and program interface level [5, 4]. As an emerging standard protocol, SOAP over HTTP is garnering a great deal of interest from industry to address the Web-based XML messaging.

## 3.2 Lupin service discovery mechanism

It aims to organize Lupin services into a coherent collection to enable discovery, i.e. locating Lupin services that provide a particular functionality and that adhere to specified constraints. 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. This is regarded to be addressed by the recommender system. 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 return the result together with the sufficient binding information to the PSE provider for the use of PSE composition. In the Web environment, XML-based technologies provide us facilities to achieve a semantic approach towards the mathematical service d escription.

### 3.3 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 be interactive to work with each other in an appropriate way. PSE composition mechanism is motivated to address this goal. In a general paradigm, the PSE builder obtains the chosen Lupin services as the "raw materials" and organizes them according to the certain application. Then, a user interface (which can be a standalone or the regular Web browser compliant) will be defined and generated as the client site front-end.

It should be noted that, from the perspective of PSE, it is not sufficient to have the only three mechanisms to define a significant PSE infrastructure. Our objective is to extract the most critical 3 parts from the whole architecture of PSE to establish the backbone, and to demonstrate whether it makes sense to shift PSE architecture to the Web environment, using the emerging XML-based technologies to support the mathematical distributed computing system.

## 4 Towards the implementation

As a Web-based framework, Lupin should be implemented by the standard Web technologies so that can be widely supported. Due to this thought, in our first attempt of implementation, we focus our eyes on *Web service*, a package of emerging technologies for program-to-program interactions.

### 4.1 Web service

The term Web service is an emerging e-business framework that can describe and interface a collection of operations that are network-accessible through standardized XML messaging [7]. The significant facilities are its potential capabilities of *service description*, *discovery* and *invocation* over the Internet offered by XML technologies. There are 3 main roles 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 registry of service descriptions 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), Web Service Description Language (WSDL), and Universal Description, Discovery and Integration (UDDI) are being developed to address its conceptual view.

### 4.2 Lupin implementation using Web service

The architecture of Web services draws us a blueprint of next horizon of e-business by its Web-based integrated mechanism and potential capabilities of service description, registration and discovery. It also 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 characteristics to the term *mathematical Web service* to support Lupin implementation. That is, our first attempt will use SOAP as the most fundamental underpinnings for XML-messaging and Web-based distributed computing to serve the PSE composition mechanism; adopt WSDL as the standard protocol for Lupin service description; and consider to use UDDI to support the implementation of Lupin discovery mechanism. For more detailed discussion about this implementation, please refer to our technical report.

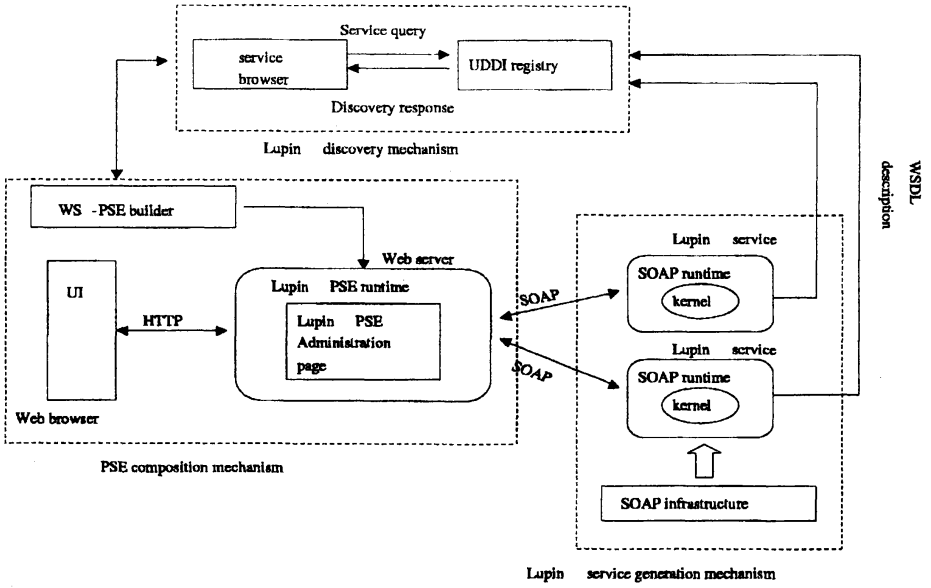


Figure 3: Lupin's implementation based on Web service

# 5 Experiments

## 5.1 Overview

This section briefly introduces our proof-of-concept system and its relevant experiments on the Lupin implementation. Highlighted works for a simple example are listed to illustrate the whole lifecycle of Lupin application. From the perspective of Lupin architecture implemented by Web service, the experiments include the process of Lupin service generation by SOAP infrastructure in the Lupin service generation mechanism; service registration and discovery by UDDI registry in the Lupin discovery mechanism; and the actual service binding and user interface by Lupin PSE runtime in the PSE composition mechanism.

Basically, the operation in the PSE composition mechanism is to allocate and remotely bind the series of discovered services in an appropriate flow and to get them working transparently for the certain class of purpose. In this experiments, our focus is on the demonstration of mathematical Web service and the exploration of how can we benefit from it. Hence we make a simply example that built on the single Lupin service: suppose one of the Lupin service is `EuclidGcd`, which accepts two integers as input and compute their GCD using general Euclidean algorithm. Fig. 4 is the whole diagram of the testing system.

Our implementation is performed under the environment of Apache Tomcat Web server. Tomcat is a highly configurable server that supports Java servlets, which can well handles the Lupin service that one wants to make accessible. On the other hand, because Lupin is designed to be a XML-based distributed environment using SOAP, among those available implementations that enable SOAP application, we chose also an Apache package called AXIS (Apache eXtensible Interaction System). Apache Tomcat Web server and AXIS construct the backbone of our experiment environment.

## 5.2 Service development, deployment and registration

The example Lupin solver EuclidGcd is implemented by a Java class `EuclidGcd`. With AXIS, it is easy to be deployed online at a certain URL, 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="Euclid1" provider="java:RPC">
  <parameter name="className" value="Euclid1"/>
```

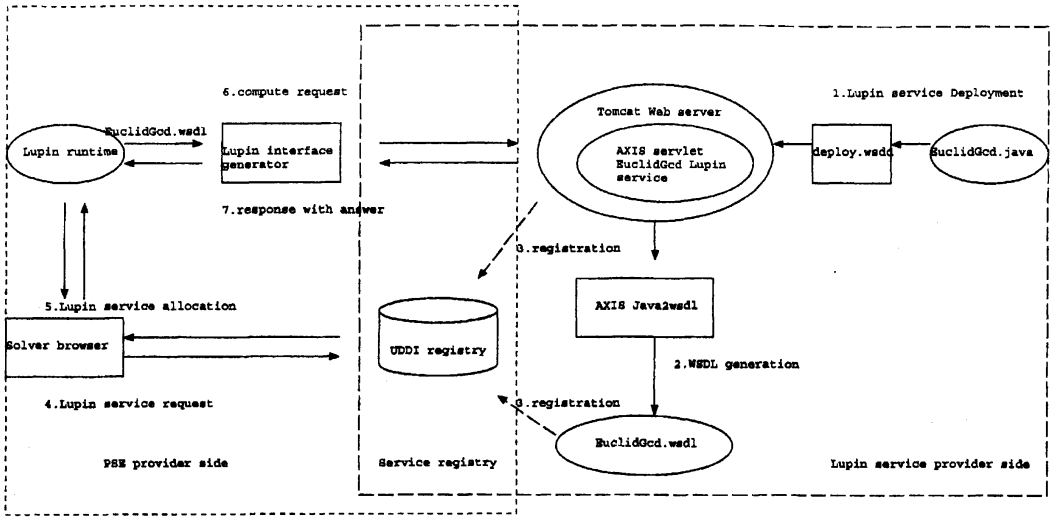


Figure 4: Operation flow of Lupin application

```
  <parameter name="allowedMethods" value="Functi"/>
</service>
</deployment>
```

The corresponding WSDL file `EuclidGcd.wsdl` can also be generated by the attached tool `Java2WSDL` provided by the AXIS SOAP infrastructure. The WSDL file, which enables the dynamic interface generation to invoke the actual Lupin service, is the key in Web service-based lupin implementation. Because the WSDL file holds sufficient information for service invocation, thus the process of discovery is essentially the process of finding the most suitable WSDL file of certain Lupin service.

When the file `EuclidGcd.wsdl` is captured by the discovery mechanism, then it can be used to be part of PSE edited by the PSE composition mechanism and to be actually invoked by Lupin PSE runtime. To make the WSDL file to be searchable, it should be firstly registered to a UDDI service registry. In our experiment, the operation is performed via systinet's WASPUDDI registry [11]. It is through the registration that some information such as the URL of `EuclidGcd.wsdl`, the service name, provider's entity and other data about Lupin service `EuclidGcd` can be stored for further search.

## 5.3 Service discovery

In Lupin, The operation of service-searching is done through Lupin service browser which is bound to UDDI registry that enables mathematical Web service discovery. Since that it is still under development now, we use systinet's WASPUDDI for experiment. According to the current UDDI specification, there are only few information can be used to perform the UDDI-based discovery such as service name and Taxonomy. Because Lupin is considered to target at mathematical application, and there is still not default mathematical taxonomy system defined in UDDI, we added a temporary mathematical taxonomy system based on GAMS for experiment. Due to the discovery facilities provides by UDDI specifications, experiments are done to find the `EuclidGcd.wsdl` that enables the further invocation.

## 5.4 User interface generation and service binding

Another important issue of our experiment is how to achieve the actual Lupin service binding when we obtained the WSDL file from UDDI registry. This operation is done by *Lupin Solver Interface Generator*, which is a part of Lupin PSE runtime and currently implemented in Java. It can dynamically and automatically generate the actual interface when given the URL of WSDL file. Fig. 6 shows how does it work: when given the URL of WSDL, an user interface is generated automatically due to the service description information involved in WSDL, and accepts the further parameter input to invoke

```
Request message:

POST /axis/services/Euclid1 HTTP/1.0
Content-Length: 495
Host: localhost
Content-Type: text/xml; charset=utf-8
SOAPAction: ""

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
SOAP-ENV:encodingStyle="http://
schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/
soap/encoding/">
 <SOAP-ENV:Body>
  <Funct1>
   <a1 xsi:type="xsd:string">54</a1>
   <a2 xsi:type="xsd:string">32</a2>
  </Funct1>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
Response message:

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 434
Date: Tue, 26 Nov 2002 08:47:12 GMT
Server: Apache Tomcat/4.0.4 (HTTP/1.1 Connector)

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
 <SOAP-ENV:Body>
  <Funct1Response SOAP-ENV:encodingStyle="http://
schemas.xmlsoap.org/soap/encoding/">
   <Funct1Return xsi:type="xsd:string">2</
Funct1Return>
  </Funct1Response>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 5: SOAP messaging during the remote computation.

the service. In this implementation, every Lupin service is sitting at SOAP runtime 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.

# 6 Remark

The experiments showed our first step on the road towards significant Lupin implementation based on Web service. Effort has been made to answer such question as: can we make use of the conceptual model of Web service and its relevant XML-based protocols to achieve our own global problem solving framework? In order to reach a satisfied answer, focus is on the two critical processes in accordance with the whole lifecycle of Lupin application: the service discovery and its invocation, which is expected to be facilitated by the protocol of UDDI, SOAP and WSDL respectively.

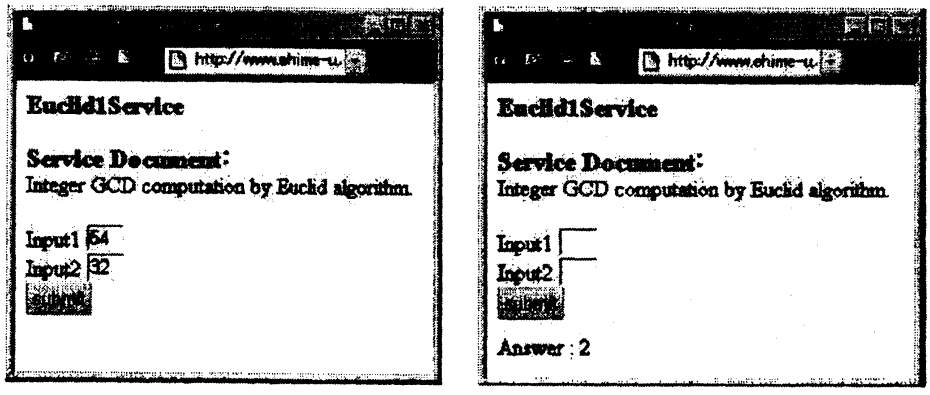


Figure 6: Interface generation and input/output from Lupin service.

According to our experiences, the process of Lupin service invocation can be sufficiently achieved by WSDL-based interface generation and SOAP-based data exchange. However, the discovery mechanism provided by UDDI registry does not completely meet our needs of Lupin service dynamic location, because of UDDI's poor description facility. From our perspective, the process of Lupin service discovery is regarded as: the process of matching a description of a service required with a description of a service advertised. Thus the process of discovery is essentially carried out by some kinds of service descriptions: one is on the base of the capabilities that offered by the service provider, while the other is about the declaration that a PSE is seeking for. From the specification of UDDI, its mechanism of description includes very little information about the content and capability of service, limiting itself to the service name, the service provider and a port where to access the service. 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 lack of mechanism to support the agent to perform a flexible and intelligent match. Due to this, a machine-understandable semantic approach which enables the agent to understand the essential relationship between the mathematical service advertisement and requirement, is needed. We are now concentrating on emerging ontology technology DAML [13], which is built on existing Web technologies such as earlier W3C standards RDF [12] and RDF schema, to extend our work. We predicate that, based on DAML ontology markup, 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 development of the description language which can be used in semantic Lupin service registration, as well as the request description. The corresponding API for Lupin registry, Lupin service browser, markup language for Lupin service flowing description and the implementation of prototypes must also be done to support the whole framework, which currently targets Web-based distributed computation and interactive mathematical education.

# 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

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

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

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

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

[12] RDF(Resource Description Framework). `http://www.w3c.org/RDF`

[13] DAML: http://www.daml.org

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