DOI: 10.15613/sijrs/2014/v1i1/53851
ISSN (Print): 2349–8919
ISSN (Online): 2350–0999
**ScieXplore: International Journal of Research in Science,** *Vol 1(1), 34–41, January–June 2014*

# Encapsulation of Micro Engineering Tools in a Co-operative Jyaguchi Computing Infrastructure

**Bishnu Prasad Gautam[1, 2*], Katsumi Wasaki[2] and Amit Batajoo[1]**

[1]Faculty of Integrated Media, Department of Integrated Media, Wakkanai Hokusei Gakuen University, Wakkanai, Hokkaido, Japan; gautam@wakhok.ac.jp, amit@wakhok.ac.jp

[2]Interdisciplinary Graduate School of Science and Technology, Shinshu University 4-17-1 Wakasato Nagano-city, Nagano, Japan; wasaki@cs.shinshu-u.ac.jp

## Abstract

Micro engineering tools and services face many issues and limitations in terms of modularity, and usability with regards to their availability and scalability. Scalability may suffer due to the limitation of service granularity as service objects sometimes may require to deliver in low bandwidth network infrastructures. Jyaguchi platform supports highly dynamic and interactive service development process by which developers can design services as per their requirement functionalities which can be micro, macro or mega services. In this paper we discuss particularly about the development process of micro services. The qualities of a micro-service are not limited to its performance metric, but are designed to maintain simplicity and may even consist of fully-functional and loosely coupled applications. These services can be utilized enabling a higher level of usability. Usability is maintained by reducing the granularity and complexity of functionalities of a service. However, Jyaguchi services have potential to increase its granularity without breaking its underlying architecture. The service oriented architecture of Jyaguchi platforms also allow for these services to dynamically deliver to the remote clients and still keep the persistent state until the client terminate the service. We also realize that micro engineering services that are distributed locally or globally should be categorized during the design phase. These services should be allowed to change its granularity in demand-respond manner based on micro, macro and mega service model instead of keeping those services static. A working model of Jyaguchi platform that can deliver these kinds of services is also discussed in this paper which incorporates all these aspects.

**Keywords:** Micro Service Engineering, Jyaguchi Cloud, Service Management, Service Modularization

## 1. Introduction

Jyaguchi is a platform at which java based application can be built and exported to the client over a network. This very means of exporting java application over a network is supported by the underlying middleware technologies such as RMI and Jini [1], [2]. Originally developed [3], [4] during his bachelor and upgraded in master degree, Jyaguchi is evolving and improving thereafter too [5], [6], [7]. Initially, the concept of Jyaguchi was brought to demonstrate how this can be exported to the client as a service. The software delivery concept was termed Jyaguchi service which literally means a tap in Japanese language. As tap can regulate the intensity or rate of flow of water, accordingly the philosophy of Jyaguchi is that user should be able to consume the service and can regulate the frequency of usage and duration of usage etc. This very nature of tap which have opening and closing valve to regulate the rate of water flow was incorporated to the philosophy of Jyaguchi architecture. Jyaguchi architecture facilitates the

*Author for correspondence

user or service consumer to consume the service as per use basis. Furthermore, Jyaguchi was developed on the concept of leasing an entire service item or application from a service provider rather than owning that software completely either by installing and licensing of software. From the novice user point of view, software installation, managing of its license, updating and upgrading of that software is a critical issue. However, providing those features as implemented in Jyaguchi reduce these kinds of management cost of the users. This sort of concept is also incorporated in SaaS based application. Jyaguchi is not only a SaaS based platform but also an architectural model that can be enhanced to model, develop and export next generation cloud based services. It utilizes the hybrid architectural model consisting of SOA (Service Oriented Architecture) and SPA (Space Based Architecture) [3], [4].

This research utilizes the Jyaguchi platform, on the top of which we built some micro-engineering tools that can be exported over networks. In academic fronts, the significance of micro-engineering tools is very demanding. In developing worlds, schools, colleges or Universities cannot afford the devices that can solve or executed the scientific problems.

This research paper reports the first phase of development of micro-engineering tools based on Jyaguchi service platform, with a particular emphasis on how its service discovery model was implemented. Service discovery model for micro-services aligns generally with the principles of ease of development, ease of deploy, ease of seek and ease of use. The paper details the ways in which the service development, its wrapping methodology, deployment and the performance metrics of Jyaguchi services. It concludes with a discussion of the transition from micro- engineering tools to the macro and mega services architecture and the performance evaluation of Jyaguchi mini service in REST and Jyaguchi platform.

## 2. Motivation and Issues

Micro engineering computing tools in computer science and engineering fields are in high demand. There is a large set of micro engineering tools ranging from simple calculation to complex calculation in the field of computer science or engineering. In this research we refer to those tools by which we can compute some mathematical problems, solve equations and plot them to the graph. Some tools even can be used to plan the network, simulate it and evaluate the performance. These tools are often provided in the market by embedding in specifically designed dedicated device. The typical example of such device is scientific calculator. There are lots of other examples such as electronic thermometers, pH gauges, weather instruments, decibel and light meters, accelerometers, sensors etc. From economic point of view, Universities and Colleges have a financial constraint to provide those tools to every student in engineering labs. In order to address such kind of financial constraint, we purpose a service development model by which such services can be provided in very cost effective manner. In this research we are trying to provide a Jyaguchi platform in which most of those micro engineering tools can be provided by encapsulating the software objects.

Jyaguchi micro-services model related to encapsulate micro-engineering tools was developed in order to re-conceptualize Jyaguchi cloud services as a set of federated computing services, developed and deployed in a decentralized manner, rather than as a centralized, monolithic manner. Though Jyaguchi provides those federated services in a monolithic service viewing browser named Universal Browser [4], the development model is not monolithic. The advantages of such an approach are that individual services are easier to maintain than the same service embedded within a larger repository framework, as well as being easier to develop, deploy and publish in the registries of Jyaguchi cloud services.

## 3. Service Development Process and its Architecture

We agree that Jyaguchi platform supports the development of distributed object. The main requirement of distributed object is its ability to create, invoke and deliver the objects in a remote host while providing the environment as if they were invoked in local infrastructure. These sorts of remote object invocation have been implemented in COM, COBRA and RMI and many other technologies until few years ago. Jyaguchi employs a similar kind of concept that invokes the total bundle of service executed in remote server. We named this invocation model as RSI. The underlying protocol to call the remote service is JERI and JRMP [8], [9]. Recently, a different type of invocation model is often utilized in web service technology such as WSIF [10]. Web service technologies has passed different stages of evolution phases in terms of utilizing underlying message passing protocols such as SOAP and REST. These technologies, must of the time, utilize XML data

format to send and receive the message. Messages can also be passed in JSON format too. However service call in web service technology and Jyaguchi is different. Most of the web service related technologies utilizes message passing technique whereas Jyaguchi utilizes RSI at which parameters are passed as the reference of java object. We did not utilize message passing rather we utilized service calling approach in order to reduce the overhead occurs in message passing. In message passing, it has to copy the existing arguments and append it to the new portion of the message resulting to a large size of message. In our approach, we used remote service invocation at which service with a unique and distinct identity, can encapsulate internal state and threads of execution, and that exhibits a well-defined visible behavior. Particularly, these services are coded as java objects and are wrapped with Jyaguchi service and the whole service is remarshaled in the user device. In the following sections, we describe the total scenarios of Jyaguchi infrastructure, service wrapping scenarios and the concept of service granularity.

## 3.1 Co-operative Computing Infrastructure

Distributed application varies in granularity and infrastructure. Traditionally, distributed applications were two-tiered, three-tiered or multi-tiered in their architecture which collectively makes a single system. This notion of single system has evolved to the creation of from tire based system to a more huge virtualized system such as grid system [11], [12]. We experienced that the emergence of cloud computing as a new platform for enterprise. From its very inception, Jyaguchi service development model utilizes the legacy computing infrastructure thereby creating a cluster of possible hardware that can participate in the federations of Jyaguchi services. The hardware and services that participate in Jyaguchi services are capable of addressing the problem in a co-operative manner. The notion of co-operation has been executed by utilizing the concept of SOA in which the underlying computing infrastructure or underlying middleware are encapsulated and the detailed of which are not required while providing the services to the end user. In fact Jyaguchi services can be built in a number of multiple technologies and protocols [13], [14]. Though there are the different architectures underlying, Jyaguchi service models can produce similar characteristics of web service and can be used together with other web service like technologies too. This ability is referred to as co-operative computing infrastructure. Figure 1 shows the relation between each device and the underlying software components that co-operate while developing, deploying and using of Jyaguchi service. In particular usage scenario, Jyaguchi client send request to the lookup server, this server provides the proxy required to the client and with the help of this proxy, client will be able to interact and can download the remaining codes from web server. In this way, a solution is achieved. In order to scale out the co-operative infrastructure, the underlying hardware federation can be increased by virtualization.

We are also exploring the ways to build the services by using encapsulation service modularization approach. We have developed complete package of middleware by integrating different kinds of underlying technologies.

## 3.2 Service Wrapping

In our test case, we developed numbers of dummy services in Java programming language in order to confirm whether the wrapper actually can work in a new environment or not. Though the architecture of Jyaguchi does not dictate or advocate any particular language, we choose this language as we have already implemented numbers of functionalities in Java. In our previous works, we had developed few services such as scientific calculator, Java Editor, Chat and Presentation Tool. These solutions are required to be wrapped by our Jyaguchi wrapper so that these objects can be published in the service registry. Figure 2 shows that each java objects are wrapped with Jyaguchi in order to publish in the Jyaguchi server.

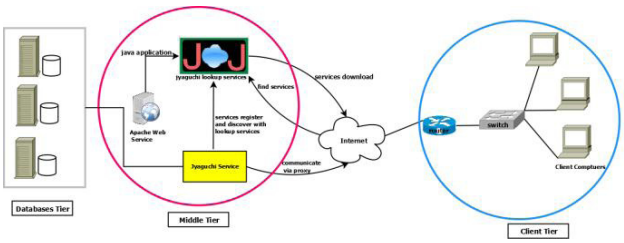


**Figure 1.** Co-operative computing infrastructure.

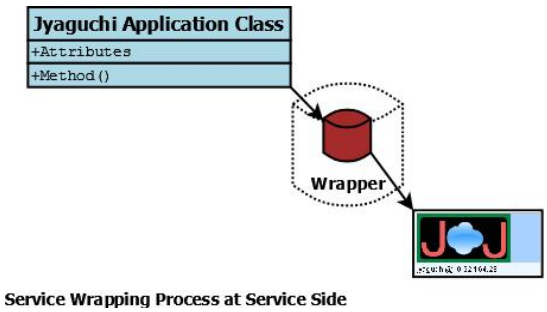


**Figure 2.** Jyaguchi Wrapper.

### 3.3 Service-client Federation

Jyaguchi platform provides services in the manner of service client federations. Figures 3 and 4 specify how Jyaguchi client can utilize Jyaguchi service in a network. In order to consume the service, service provider must publish his service in a network. Generally, this Jyaguchi service is published in lookup server. If the client wants to utilize this service, he or she must have the account. User accounts and other service status information are stored in database. While client decides to utilize the service, he or she must login into his account. Published services in service registries can be browsed by using universal service browser provided by Jyaguchi development package. In order to consume the service, user needs to pass the authentication process. After successful authentication, he or she can utilize the service by simply clicking the icons displayed in the browser. In order to complete this process, service and client have an interaction for number of times.

## 4. Modularization and Encapsulation of Services

Modern ICT companies and organization are facing number of problems while they want to respond to rapid market demands. In order to respond to the rapidly changing market, organization often tries to participate and collaborate with inter-organizational networks and solutions. By participating into inter-organizational networks, they are able to provide new kind of services and solutions but face a number of challenges. These challenges occur due to sheer size and inter-mixed solution which has no proper documentation to further scale up. In order to solve such complexity, the modularization approach has gained lots of attention from the researcher these days. In the past decade, we witnessed that most of the ICT solutions are architected in SOA [15], [16], [17] principle. As SOA makes it possible to encapsulate underlying business process, it has been widely applied in business applications architecture. Furthermore, it also makes possible to introduce newer concept of ICT enterprises such *as* service bus, service composition, and service virtualization. In order to retain the qualities of SOA, it is inevitable to encapsulate the complex functionalities into a separate bundle of the software. The process of separating the functionalities from different features of remaining functionalities is called modularization. We got back into

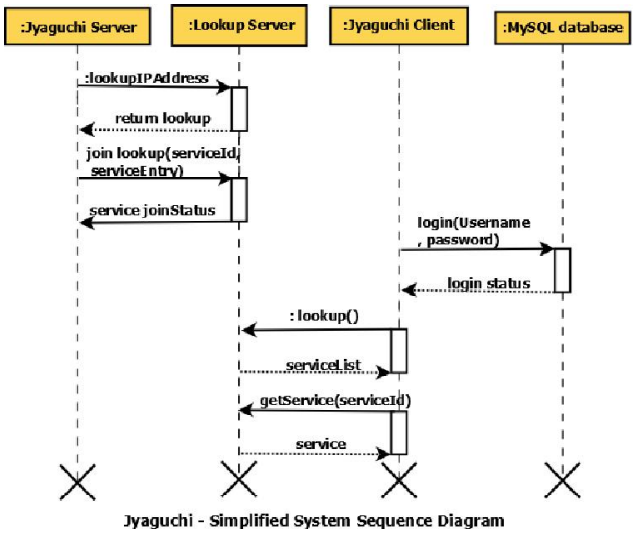


**Figure 3.** Scenario between Jyaguchi Service and Jyaguchi Client.

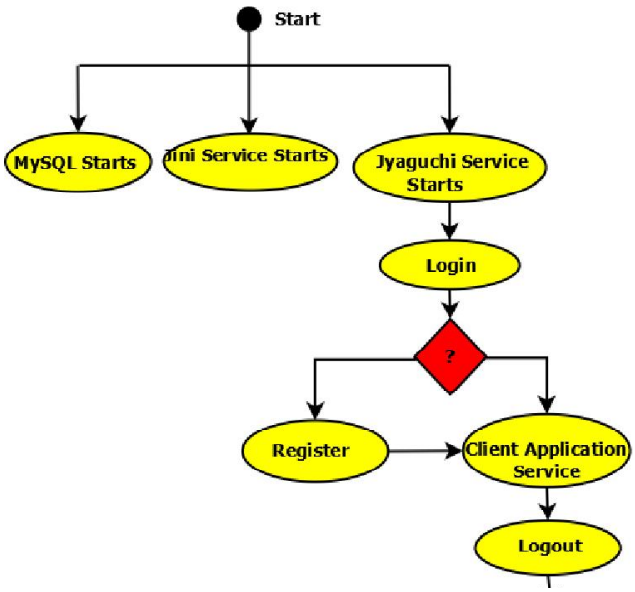


**Figure 4.** Activity diagram of Jyaguchi Service and Client.

Jyaguchi research after a few gap of times in order to make service modularization more explicit.

This paper provides insights into the concept of service encapsulation as a process of modularization. To define more properly, service modularization is the act of providing a solution as a module creation. When the service modularization can be started by encapsulating the functionalities into number of different modules, then it is also

equally important to modify non-modularized services to modularized ones that can leverage entire encapsulation process. So how do we encapsulate the non-modularized process into a single module?

First, we sought to categorize the existing services of Jyaguchi and decompose into finer grain. In order to provide basic and simple mathematic calculation, we tried to implement a single method for a single functionality. Such kind of simple function is wrapped by Jyaguchi wrapper. This is the core feature required in SOA based modularization. In enterprise, IT assets have to be wrapped and provide interface to access these assets. These assets can be ranged from a simple program or a simple hardware to a large piece of software and hardware. In this paper, we limit our modularization just to a software service. Jyaguchi services also ranged from very simple program to large piece of software. As the granularity of this service is small, we name it micro service our modules at this stage completely based on software systems only and we have not included humans in the modularization process. We introduce about these modules in the subsections below (Refer subsection 4.1, 4.2 and 4.3).

The concept of modularization is to separate the concern and context from once piece of program to other so as to minimize the effect that changes in one module may have on other modules. Separating the unrelated concerns from the modules has a great advantage of reducing interdependencies of modules so as to minimize the coupling between the server and client program. While there is a maximum coupling between server and client program, a small change in server program needs to be informed to the user. While this sort of software cohesion and coupling issues can be addressed in service modularization [18], [19]. In the following subsections, we describe our guideline indicator about the notion of services and we categorized the services into 3 different sub-groups.

### 4.1 Mini Service

Mini services in Jyaguchi are the service which can be downloaded over a network and these services can be exported as a complete software package. To utilize this service, Jyaguchi client even does not require knowing about the interface. However Jyaguchi client must possess universal browser, where we have implemented universal interface that can be used to call any Jyaguchi mini service. These services are implemented for the users who have low internet bandwidth.

### 4.2 Macro Service

Macro services are the services which can either be downloaded or can be accessed via web browser. As the size of service becomes larger than micro service, we categorized these kinds of services to macro service and put option to the client to choose whether he or she wants to download entire service or can accessed by using web browser as like other web services.

### 4.3 Mega Service

Mega services in Jyaguchi are those services which granulairty is extremely larger than macro services and are not feasible to serialze as a comlete software that can be done for micro and mega services. Option is also omiitted and client can only use this kind of service as like web services by using web browser.

## 5. Evalaution of Modules

In this section, we describe about evaluation of modules that we created during our research. In our previous studies [3], [4], [6], we have implemented Jyaguchi Editor, Jyaguchi Calculator, Jyaguchi Presentation and few other modules. These modules are continuously upgraded and we have added new functionalities too. During this research, we presented new notion of modularization into the previously developed services. In order to simplify our explanation in this paper, we would like to report our result of Pi value calculation. This function was modularized in the application of Jyaguchi Calculator. We implemented this service in two different platforms: REST [20] and Jyaguchi.

### 5.1 REST vs. Jyaguchi Service

Figures 5 and 6 are the detailed benchmarked result of Pi calculation observed in REST and Jyaguchi platform. Figure 5 a), b) and c) were taken while this service was executing by using REST, whereas Figure 6 a), b) and c) were taken for Jyaguchi platform. Our result showed that garbage collection in Jyaguchi is far effective than in REST platforms. For the rest of performance evaluation we discuss in the next section.

#### *5.1.1. Analysis and Discussion*

Figure 5 a) and Figure 6 a) show the consumption size of heap after executing 1000 loops for pi value calculation.

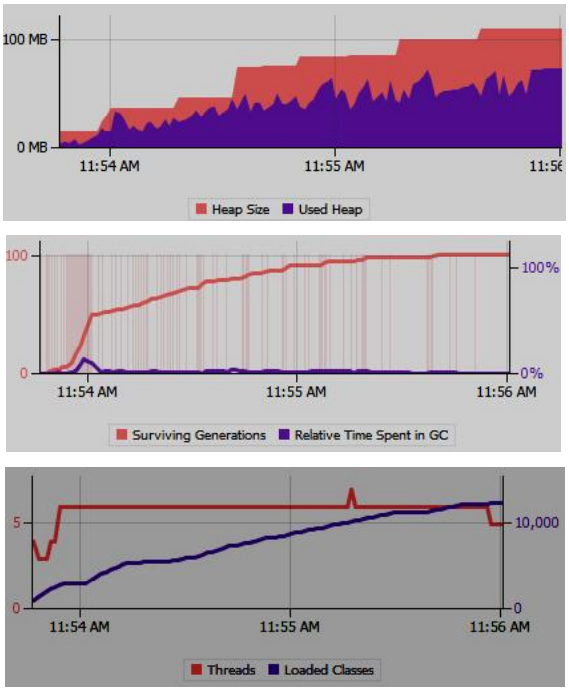


**Figure 5.** a) Heap consumption in REST b) Garbage collections status in REST c) Loaded class and threads in REST

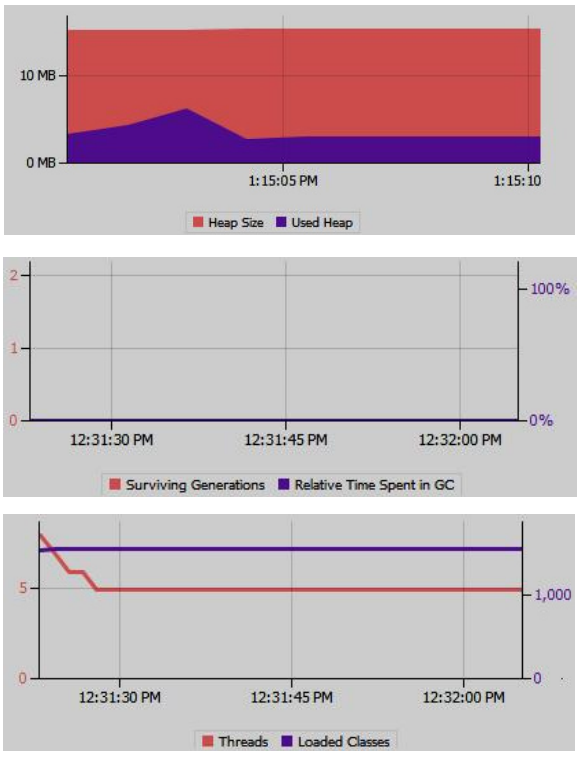


**Figure 6.** a) Heap Consumption in Jyaguchi b) Garbage Collections Status in Jyaguchi c) Garbage Collections Status in Jyaguchi

The value was resulted with 50 decimal points. The calculated value (3.14059265383979292596359650286939 5970451389330779 84) was same in the both platform of REST and Jyaguchi. As you can see, there are different graph between REST and Jyaguchi. Initially, the size of heap consumed in REST is small and it is in increasing trend because in REST platform, initially, HTTP request is sent to the server and there is no much heap consumption, however after receiving the response from the server, the client side need to re-serialize the obtained message. Though REST supports both XML and JSONObject, in our experiment, we used JSONObject. From above graph we can see that garbage collection need to wait pretty long until the object is re-serialized in REST. Instead, in Jyaguchi this happened relative quickly (See Figure 6 a) and Figure 6 b). The rest of figures are self-descriptive and we do not find much difference. However, in our observation, REST will take more memory and more time to re-serialize data if the size of message is increased. The benefit of REST is it does not dictate client to use JAVA or any specific environment except simple browser, where Jyaguchi requires java environment for current implementation.

## 6. Future Works

Jyaguchi is continuously evolving research project. The author [3], [4] started this project in 2002 and have upgraded and utilized by numbers of peoples [6], [7]. Recently, we began to implement micro-engineering tools and tested whether we can utilize those tools in the LAN without having difficulties. We did not have much difficulty to implement Jyaguchi services particularly the micro-engineering services and expose them via look up service for end users. We believe that end users will have no problem to utilize micro-engineering Jyaguchi tools in their environment. However, we realized that we need to do sufficient works to implement mega service that can be utilized by Jyaguchi client.

In this paper, we have presented a very simplified model that explains how to integrate heterogeneous environments as a co-operative computing infrastructure. These co-operations may include interaction between backend servers that may use relational database server, NoSQL, service registry, web server, legacy server and many other computing resources. We have illustrated the model in the context of simple mathematic calculation example. We have also shown how to develop, deploy and use Jyaguchi services.

While evaluating our numbers of micro-services, we identified a number of issues for further research. For instance, we have to limit the size of service and identify what sort of service suit to utilize the services provided by this system. Furthermore, we have to provide a framework by which development of each set of micro, macro and mega services can be plugged in our system smoothly. We also would like to develop a workload partitioning or balancing packages with more precise partitioning function. We would like to deploy these services in the networks of Himalayan regions [21], [22] that the authors have constructed in their other research projects. Also, we need a full-fledged server side administration package to properly manage our deployed services.

## 7. Conclusion

In this paper we highlighted about our lab experiment regarding the advantages of using Jyaguchi platform while you export encapsulated engineering tools or services. We argued that in Jyaguchi, Java programming language is playing very important role to create dynamic networking computing system and a distributed service-oriented programming model that forms a strong base for cloud services. We presented an overview of Jyaguchi system that aims to demonstrate the benefits of service encapsulation. We also calculated pi value and compared the performance between Jyaguchi and REST based infrastructure. Our result showed the mixed result however, the memory usage in Jyaguchi platform is more stable and effective. Furthermore, we defined the granularities of Jyaguchi services in terms of Micro, Macro and Mega services.

## 8. Acknowledgement

The authors would like to thank the entire family of Wakkanai Hokusei Gakuen University and Shinshu University for their support.

## 9. References

1. Oracle Documentation about Java RMI tutorial. Available: http://docs.oracle.com/javase/tutorial/rmi/overview.html
2. Wollrath A., Riggs R., and Waldo J., "Sun Microsystems, Inc, A Distributed Object Model for the Java System". Available: http://pdos.csail.mit.edu/6.824-2009/papers/waldo-rmi.pdf Date Accessed: 2014/05/25
3. Gautam B. P., "An architectural model for time based resource utilization and optimized, [Master Thesis], Shinshu University, 2009.
4. Gautam B. P., and Shrestha D., "A model for the development of universal browser for proper utilization of computer resources available in service cloud over secured environment", *IAENG 2010*, Available: http://www.iaeng.org/publication/IMECS2010/IMECS2010_pp638-643.pdf
5. Kleinberg J., "Complex networks and decentralized search algorithms". Available: http://www.cs.cornell.edu/home/kleinber/icm06-swn.pdf
6. Shrestha S. K., Kudo Y., Gautam B. P., and Shrestha D., "Multidimensional service weight sequence mining based on cloud service utilization in Jyaguchi", Available: http://www.iaeng.org/publication/IMECS2013/IMECS2013_pp301-306.pdf
7. Shrestha S. K., Kudo Y., Gautam B. P., and Shrestha D., "Recommendation of a cloud service item based on service utilization patterns in Jyaguchi", *Proc of KSE2013*, vol. 2, pp. 121–133
8. Newmarch J., "Foundations of Jini 2 programming". Available: http://jan.newmarch.name/java/jini/tutorial/Jeri.html#Jeri
9. Apache River Project. Available: http://river.apache.org/doc/specs/html/lookup-spec.html
10. Gautam B. P., Sharma N., and Wasaki K., "Using a solar powered robotic vehicle to monitor and manage unstable networks", *ICFN 2014*.
11. Gautam B. P., Sharma N., Shrestha S., and Gautam R., "Monitoring and management of unstable network through solar powered robotic vehicle", WAKHOK University Bulletin, vol. 14, pp. 19–30, Mar. 2014.
12. Tuecke S., Czajkowski K., Foster I., Frey J., Graham S., Kesselman C., Maquire T., Sandholm T., Snelling D., and Vanderbilt P., "Open grid services infrastructure", Available: http://toolkit.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf
13. Chen M. X., Sung F., and Lin B. Y., "Service discover protocol for network mobility environment", *ICIC International Journal 2012*, Available: http://www.ijicic.org/ijicic-11-05025.pdf
14. Perianu R. M., Hartel P., and Scholten H., "A classification of service discovery protocols". Available: http://doc.utwente.nl/54527/1/classification_of_service.pdf
15. van der Aalst W. M. P., Beisiegel M., van Hee K. M., König D., and Stahl C., "A SOA-based architecture framework", Available: http://www.win.tue.nl/~cstahl/Papers/AalstBHKS2007_csrep.pdf
16. Service Oriented Architecture (SOA) and Specialized Messaging Patterns, Technical White Paper, Available: http://www.adobe.com/enterprise/pdfs/Services_Oriented_Architecture_ from_Adobe.pdf

17. Sahin K., and Gumusay M. U., "Service Oriented Architecture (Soa) based web services for geographic information systems". Available: http://www.isprs.org/proceedings/XXXVII/congress/2_pdf/5_WG-II-5/03.pdf
18. Terlouw L., "Modularization and specification of service-oriented systems", Ph.D Thesis, 2011.
19. Brax S. A., and Toivonen M., "Modularization in business service innovations", Available: www.imi.tkk.fi/publications/download/207/
20. Fielding R. T., "Representational State Transfer (REST)", Ph.D Thesis, 2000, Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
21. Paudel D. R., Kazuhiko S., Gautam B. P., and Shrestha D., "Design and implementation of partial mesh community wireless network in himalayan region", *Proceedings of Third Asian Himalayas Internationl Conference on Internet AH-ICI2012*, 2012/11, Kathmandu, Nepal.
22. Gautam B. P., Paudel D. R., and Shrestha K., "A study and site survey in himalayan region for proper utilization of wireless community networks: an assessment of community wireless implementation in heterogeneous topography", *WAKHOK Journal*, vol. 11 Japan Disaster Statistics, Natural Disasters from 1980–2010, International Disaster Database. Available: http://www.preventionweb.net/english/countries/ statistics/?cid=87