

# Converting Legacy Application to Web Services using an Open Source Toolkit: A Case Study

Rahul Upakare, M Sasikumar  
Open Source Software Resource  
Centre, CDAC, Sector 7, Raintree  
Marg, CBD Belapur, Navi Mumbai,  
400 614, India, +91-22-27565303  
{rahulu, sasi}@cdacmumbai.in

## ABSTRACT

Today a large number of software tools are available to use computer to support traditional on-campus based courses. One such tool that we are using for automated program grading in our on-campus diploma programmes is Parikshak system. Today many Learning Management Systems (LMSs) are available for delivering courses online or in blended mode. Tools which are useful for on-campus course students can be easily integrated with the chosen LMS, to make them available for off-campus students. But, most of the times, these tools (like Parikshak) gets implemented as a standalone application without any inherent mechanism to allow seamless integration with other applications or frameworks. In this paper, we investigate how well Web services meet the interoperability requirements by transforming a real example of Parikshak system into a program grading Web services framework using the open source gSOAP toolkit and providing standard Web service interfaces for seamless integration to other applications.

## Keywords

Program grading, Web services, Interoperability

## 1. INTRODUCTION

Programming assignments are a critical part of programming courses. It is difficult and time consuming to evaluate programming assignments manually, because of the complexity and sheer numbers of students involved in such courses. It is also difficult to ensure uniform standards of evaluation. To address such problems, we have a system called Parikshak – an automated program grading system - which provides a secure and restricted development environment for students to develop and submit their programming course assignments and to receive immediate feedback. It provides course staff tools to monitor and control the process and the ability to generate student assignment submission reports to track students performance.

Today many Learning Management System (LMS) are available for delivering courses online or in blended mode. Tools such as Parikshak need to be integrated with the chosen LMS, to make them accessible seamlessly under a single environment. But, most

of the times, these tools (like Parikshak) gets implemented as a standalone application without any inherent mechanism to allow seamless integration with other applications or frameworks.

Web services is an emerging technology and useful to implement Service Oriented Architecture (SOA) [4]. Web services offer a strong foundation for software interoperability through the core open standards of eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Description, Discovery and Integration (UDDI). SOAs are transforming monolithic applications into services, thus allowing portability, reusability and on-demand access. Instead of thinking about applications residing on a specific server, we should think of ubiquitous services that can be used and shared more dynamically. The implied benefits are simplified design, code reuse and a giant step toward "business agility" [3]. SOA solutions are composed of reusable services, with well-defined, published and standards-compliant interfaces. SOA can be used as a mechanism for integrating existing legacy applications regardless of their platform or language.<sup>1</sup>

Concepts like Service based software [2], Software as a Service [1], etc. are emerging rapidly and has given innovative vision for future software architecture and development. Software as a service (SaaS) is a software distribution model in which applications are hosted by a service provider and made available to customers over a network, typically the Internet. SaaS is becoming an increasingly prevalent delivery model as underlying technologies that support Web services and SOA mature and new developmental approaches, such as Ajax, become popular. There is interest in moving many applications from their existing framework to run over the web under a Web services framework.

Parikshak is currently a standalone system in the typical legacy application mode. We believe that integration of such a legacy software, into a LMS will be smoother if it is transformed into a Web service. Further we believe that Parikshak Web service is best seen – from machine perspective – as a collection of services, each service performing a specific task. Such an approach enables many natural enhancements to Parikshak and also provides many advantages, as discussed in subsequent sections.

The remainder of this paper is organized as follows. Section 2 discusses the Parikshak system which is being used at our campus for automated program grading. The open source gSOAP toolkit is briefly discussed in Section 3. Section 4 discusses Parikshak and Web services and our phased approach to transform Parikshak

---

<sup>1</sup> Quotes from <http://serviceorientation.org/>

into a program grading Web services framework. In section 5, we discuss the program grading APIs, Conclusion and future work is discussed in section 6.

## 2. ABOUT PARIKSHAK

Parikshak is a system for testing problem solving skills through programming, which provides a restricted environment for students to develop and test their program. Parikshak is a legacy system and has evolved from bunch of shell scripts to a collection of C programs. It has been built and maintained in-house. Our centre and affiliates are using Parikshak system in programming modules of our diploma programs to automate the program grading.

### 2.1 Parikshak – A User Perspective

Parikshak is used in two modes: assignment and evaluation. Assignment mode is used to solve the assignments and evaluation mode for programming tests. We use Parikshak for Object Oriented Programming using Java (OOPJ) and Data Structures and Algorithms (DSAL) modules of our post graduate diploma programmes.

Parikshak Administrator can install assignment package for a particular course module. An assignment package contains one or more programming assignments to be solved by students. Students have to submit these assignments within the specific period set by the Administrator. Parikshak keeps log of all the submissions done by students and does not allow any submission after or before the specified period.

Parikshak is used in the evaluation mode for conducting a Machine Graded Programming Test (MGPT) [6]. MGPT is conducted in two sessions. The first session is conducted in a classroom under manual supervision. In this session, students get a problem statement to read, understand and develop the solution on paper along with suitable test cases. Access to computer facility is not allowed during the first session. In the second session, students use computer to invoke Parikshak in the evaluation mode and then they can write and submit the solution for grading. Students get limited time during both the sessions.

From a user (student) perspective, Parikshak environment is essentially a restricted Linux shell. Many of the Linux commands have been disabled in this environment from examination security point of view and to discourage copying, etc. and some additional commands have been provided. The Parikshak environment is invoked, using the command *testme*. Command *showproblem* shows the problem statement. Users can use *vi* or *nano* editor to write the program solution and can submit it for testing using *submit* command. When the program is submitted for testing, Parikshak will compile the program, listing the compilation errors if any. If the program solution compiles successfully, the system will run the submitted program and give it test inputs and evaluate the output.

### 2.2 Parikshak Architecture

Parikshak system is not a single executable. It is a collection of C program executables. Each of these executables act as a module to provide the specific functionality. The major modules and their relation to various commands given earlier are shown in Figure 1.

The different modules include assignment and test packing module to create the installable package, assignment and test unpacking module to install the installable package, users module to control the access, and report generation module to generate reports.

Here, we attempt to reorganise this Parikshak system to make it interoperable with other systems using web services based approach.

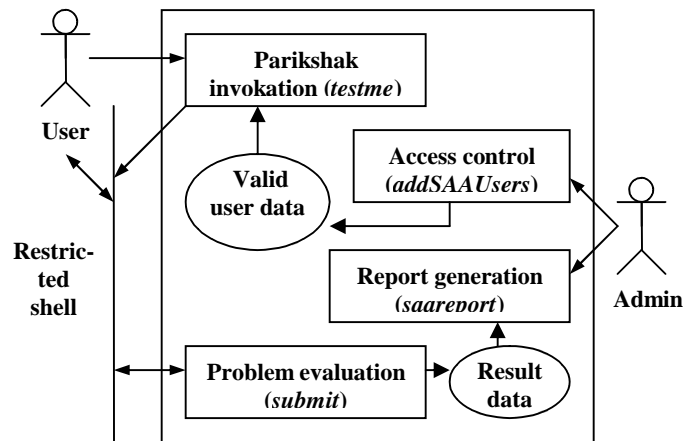


Figure 1. Parikshak as collection of executables

## 3. INTRODUCTION TO OPEN SOURCE gSOAP TOOLKIT

The gSOAP compiler tools provide a SOAP to C/C++ language binding to ease the development of Web services and client application in C and/or C++. gSOAP is one of the few SOAP toolkits that support the full range of SOAP 1.1 RPC encoding features including sparse multidimensional arrays and polymorphic types [5]. For example, a remote method with a base class parameter may accept derived class instances from a client. The gSOAP toolkit also supports streaming DIME attachment transfers, which allows exchange of binary data. The gSOAP 2.0 and higher is thread safe and supports the implementation of multithreaded standalone services in which a thread is used to handle a request.

To offer application as Web service, it must implement a set of SOAP-compliant RPC functions to expose the service on the Web for remote invocation. The gSOAP toolkit completely automates this task. The gSOAP compiler can also generate a WSDL document describing the service in detail; this would be useful to discover the Web service.

The deployment of a Web service as a CGI application is one way to provide service on the Internet. gSOAP services can also run as standalone services on any port by utilizing built-in HTTP and TCP/IP stacks. We have used gSOAP toolkit to implement our system, thanks to the strengths mentioned above.

## 4. PARIKSHAK AND WEB SERVICES

This section discusses challenges and issues in transforming Parikshak into Web services, and a phased approach to transform Parikshak, a legacy standalone program grading system, to a program grading Web services framework.

Web services are self-contained applications that are published, located, described and invoked over the Internet or intranet. A Web service can be built from the ground up as a new application or an existing legacy system can be re-engineered to make it Web service enabled. Offering application like Parikshak as a Web service will help to access it not only through the Internet, but will also make it platform and programming language independent. This means the Parikshak service invocation can be easily integrated with any Learning Management System like “Moodle” written in *PHP* or Integrated Development Environment like “Eclipse” written in *Java* and can be accessible transparently by the users. A Web service model widens the reach of Parikshak to other devices which can support development of Web services clients like Hand-held devices, mobile handsets etc.

Conversion of Parikshak into a Web services framework is achieved in phases. In phase 1, programmatic access to user/student related functionalities of the system is provided. In phase 2, Parikshak is reorganised into a collection of services, and finally a service discovery component is added in phase 3.

### 4.1 Challenges and Issues

Before designing Parikshak as a Web service, let us find out the challenges and design issues involved in implementing the Program grading Web service.

#### 4.1.1 Long response time

For every program grading request, Parikshak service requires compiling and grading the solution against 6-8 pre-defined test cases. Therefore the response time for handling program grading requests by the service will be long. This means that if the service is serving the program grading request from one client, other clients may not be able to use Parikshak Web service at the same time.

#### 4.1.2 Time restrictions

When we use Parikshak for Machine Graded Programming Test (MGPT) [6] students get 90 minutes on computer in the second session to write and submit the solution for grading to Parikshak as described in section 2.1. Such time restrictions needs to be imposed fairly and consistently by Parikshak program grading Web services framework also.

#### 4.1.3 Malicious code

The program grading service compiles and executes the source code submitted by the user. Since, program will be getting executed on the server, there is a possibility that someone can try to submit a malicious code to bring the service or server down. For example, calling fork system call to create a new process infinitely on Unix machine, can bring down the server. Also, one can try to delete or modify the existing files.

#### 4.1.4 Interoperability

Web services technology can provide a new level of interoperability between software applications. There are many

frameworks are available to enable software with SOAP, WSDL, and UDDI capabilities. But, not all frameworks are yet interoperable with each other. Also, these frameworks might support different version of W3C recommended specifications for SOAP, WSDL etc. Also, to ensure the interoperability, it is important to decide the data types of parameters which will be used in Web service function calls.

### 4.2 Phase 1: Program Grading Web Service APIs for Parikshak

In phase 1, we have enabled programmatic access [7] to Parikshak including problem set enquiry, problem set report etc. For this, we have made all student related functionality of the system available as program grading Web services APIs, which enables seamless integration of Parikshak with other application just by implementing a Web service client.

Figure 2 shows Parikshak transformed, this way, into a Web service for a student/user. A service has been written as a multithreaded standalone server. A client application can be implemented to invoke the Parikshak functionalities using the provided APIs. W3C recommended Standard Object Access Protocol (SOAP) is used for messaging.

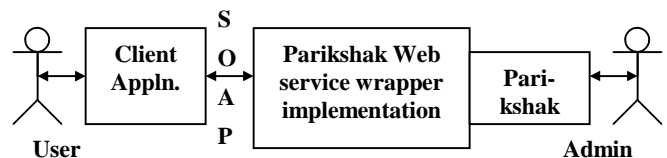


Figure 2. Parikshak System as a Web Service

Web services framework is implemented as a wrapper over existing Parikshak system, with some of the main components of Parikshak re-implemented to suit the Web services model. Using the WSDL document of the service, the service consumer can get the details of the data types definitions related to the messages being exchanged, messages name, parameters, message binding, and service location. The user then can send appropriate request using those APIs. The APIs allow user to request for available problem set names, details about any particular problem set, a particular problem statement within a set, etc. Users can write a solution for the selected problem and send it for evaluation to Parikshak. Also, they can send request to get the problem submission status for a given problem set.

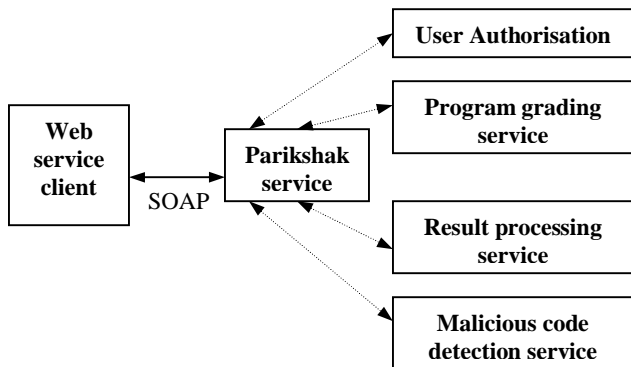
A Web service client can be written in any programming language and on any platform using these APIs.

### 4.3 Phase 2: Conversion of Parikshak into a Collection of Services

While a human user tends to perceive Parikshak as a single well-integrated service, we believe it is best seen – from a machine perspective – as a collection of services, each service performing a specific task.

In Phase 2, we separate the independent functionalities within, as independent Web services. As we can see in Figure 3, user authorisation, program grading, result processing and malicious

code detection are all almost independent functionalities within Parikshak. Therefore, we can view Parikshak as a loose coupling of such functionalities by implementing those as separate Web services.



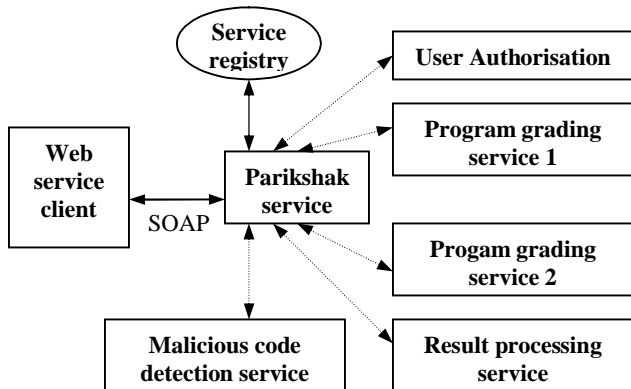
**Figure 3. Parikshak system as a collection of loosely coupled services**

For external user, it continues to be a single interface i.e. a Parikshak service, but internally it will be a collection of services loosely connected with each other. In fact, due to this loose coupling, service implementations can be scaled, evolved and changed dynamically, as long as each continues to implement the specified service contract.

Also, in case better service for any of the components is available elsewhere providing the matching functionality, then it can be integrated with Parikshak seamlessly. This also enables Parikshak to run across multiple platforms with the core engine running on one platform, and the compiler/evaluator for different languages running on different machines under different operating systems. Each of the component services can also be further specialized into another level of services e.g. Java malicious code detection service, C malicious code detection service, etc.

#### 4.4 Phase 3: Web Services Oriented Architecture for Program Grading

In phase 2, we have seen Parikshak as a collection of loosely coupled services. Here, we extend the organisation of Parikshak as a collection of loosely coupled services which are discoverable using a local service registry as shown in Figure 4.



**Figure 4. Service oriented architecture for Parikshak**

A service consumer need not have to worry about a particular service location. Parikshak Web service can make an appropriate choice on behalf of the consumer using the service registry.

The service registry is implemented as another Web service. The Parikshak service can invoke the service registry APIs to get the appropriate location of the User authorisation service, Program grading service, etc. Currently, service registry can provide a service location for requested service. It also allows service provider to publish and un-publish their services.

### 5. OBSERVATIONS

During the implementation of all three phases, we found the gSOAP toolkit very useful. To address the long response time issue as discussed in section 4.1.1, we have implemented Parikshak service as multithreaded server. The gSOAP supports implementation of multithreaded standalone services and we found it very easy to implement.

As shown in Figure 3 and Figure 4, Parikshak Web service acts as a service for Web service clients and as a client to program grading service, malicious code detection service, service registry etc. Implementing client functionality within the Web service can be easily achieved using the gSOAP toolkit.

Web services are stateless and as discussed in section 4.1.2, program grading service requires maintaining the time restrictions. We address this issue, by implementing our own session management. When a user requests a problem statement, service adds entry to session list and returns a session key. A user needs to send this session key for all subsequent calls.

We did not face any problem in deciding the data types of messages as mostly they were of type strings or integer. Only in case of problem statement file, we have used base64 binary format to support statement file in various formats like PDF, DOC etc. Also, gSOAP support the use of XSD type encoded messages to improve the interoperability.

### 6. PARIKSHAK WEB SERVICE APIs

In this section, we briefly describe some of the major service APIs that we have implemented as part of the phase-1 transformation.

#### 6.1 Program Grading Related Requests

The student/user needs to communicate with the service to get the problem statement, submit the solution for evaluation, getting the problem submission status for a set etc. Some of the requests provided for student/user are listed below.

##### 6.1.1 Problem Set Names Request

As the name suggest, this requests problem set names to Parikshak program grading Web service and receive in return an array of problem set names.

##### 6.1.2 Problem Set Details Request

Problem set details request submits a name of problem set to Parikshak program grading Web service and receive in return the list of problem number and corresponding problem title, in the requested problem set.

### 6.1.3 Problem Statement Request

Problem statement request submits a name of problem set, a problem number and expected problem statement file format (text/html, text/plain, text/pdf etc.) and receive in return the problem statement for the requested problem number in selected problem set and in the specified file format. The problem statement is sent as base64 encoded text. This request also returns a session key which user need to send with every further request (regarding this problem) to the program grading service.

This request has two variants, one for assignment mode and the other for test mode.

### 6.1.4 Compile Problem Request

Compile Problem request submits a source code of implemented solution and programming language to Parikshak program grading Web service and receive in return the compilation status of the submitted solution or listing of compilation error if any.

### 6.1.5 Submit Problem Request

Submit Problem request submits a user ID, problem set name, problem set number, session key and source code of implemented solution to Parikshak program grading Web service and receive in return the grading status of the submitted problem or compilation error if any.

Also, there is *Get Remaining Time Left Request* to check remaining time and *Get Problem Set Submission Status Request* to check problem submission status.

## 6.2 Other Requests

The other requests include requests which are sent among the Parikshak program grading service and the individual web services mentioned in phase 2 or which are not specifically related to program grading functionality. Some requests like *get problem set report request* and *is malicious code request* can be invoked directly by service consumer, provided the service details are known and accessible to Web service client.

### 6.2.1 Get Problem Set Report Request

Get problem set report requests submit a problem set name to result processing service and receive in turn the problem set report containing detailed information about student problem submission status for selected set. This request is restricted to Parikshak administrator only. Currently HTTP authentication is used to authenticate the Administrator in addition to secret registration key security.

### 6.2.2 Is Valid User Request

Is valid user request submits a user ID and registration key and problem set name to authorisation service and in turn receive true or false based on whether the user is valid or not, for the particular assignment/test.

### 6.2.3 Get Service Location

Get service location request submits a name of service and in turn receive the location of requested service.

## 7. FUTURE WORK

As discussed in our phased approach, we have successfully implemented all the three phases. As a future work we will implement multiple instances of the various component services running to share the load among the available program grading service instances. This multiple service provider idea can be used for any service which takes longer response time or generally gets loaded heavily. This also helps to make sure the high availability of services, because if one service provider is not available then the request can be passed to other available provider offering similar service. This means there are less chances of total service failure. We also plan to convert more of the legacy components of Parikshak into a more Web service complaint structure. These two will significantly enhance the scalability and extensibility of the Parikshak system.

## 8. CONCLUSION

In this paper, we have described our approach to reorganise Parikshak under a Web service based service oriented architecture to enhance its accessibility and scalability and to make it interoperable with other tools. As discussed in section 4, we are aiming to develop a complex service oriented architecture. Our choice of open source gSOAP toolkit for the implementation proved successful. The gSOAP stub and skeleton compiler provides a good SOAP to C++ language binding for development of SOAP enabled Web services and clients. Overall our experience with Web service technologies when developing the all the three phases and planning the system evolution has been positive.

## 9. ACKNOWLEDGMENTS

This work was carried out under the Open Source Software Resource Centre (OSSRC) project, which is a joint initiative of IBM, Indian Institute of Technology Bombay, Mumbai (IITB) and Centre for Development of Advanced Computing (CDAC). We also thank OSSRC colleagues for their valuable comments and suggestions on this paper.

## 10. REFERENCES

- [1] Turner, M., Budgen, D., and Brereton, P. Turning Software into a Service. *IEEE Computer*, (Oct. 2003), 38-44.
- [2] Bennett, K. H., Layzell, P. J., Budgen, D., Brereton, P., Macaulay, L., and Munro, M. Service-based Software: The Future for Flexible Software, *Proceedings of 7th Asia-Pacific Software Engineering Conference*, IEEE Computer Society Press, (Dec. 2000), 214-221.
- [3] Foster, I. Grid's place in the service-oriented architecture, [http://www.computerworld.com/softwaretopics/software/apps/story/0,10801,97919,00.html?from=story\\_kc](http://www.computerworld.com/softwaretopics/software/apps/story/0,10801,97919,00.html?from=story_kc), November, 2004.
- [4] Thomas Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall, 2005.
- [5] Robert van Engelen, *gSOAP User Guide*, <http://www.cs.fsu.edu/~engelen/soap.html>, February, 2006.

[6] *CDAC Mumbai course Handbook*,  
<http://www.cdacmumbai.in/education/coursehandbooks/2005/>, 2005.

[7] Petinot, Y., Giles, C.L., Bhatnagar, V., Teregowda, P. B., Han, H., and Council, I. A service-oriented architecture for digital libraries, ICSOC, (Nov. 2004), 263-268.