A gentle transition from Java programming to Web Services using XML-RPC

Marc Conrad - Tim French
Department of Computing and IT
Faculty of Creative Arts and Technology
University of Luton, LU1 3JU, UK
Marc.Conrad@luton.ac.uk, Tim.French@luton.ac.uk

Abstract
Exposing students to leading edge vocational areas of relevance such as Web Services can be difficult. We show a lightweight approach by embedding a key component of Web Services within a Level 3 BSc module in Distributed Computing. We present a ready to use collection of lecture slides and student activities based on XML-RPC. In addition we show that this material addresses the central topics in the context of web services as identified by Draganova (2003).

1. Introduction

The continuing global expansion of various forms of e-commerce as exemplified by B2C (Business to Consumer), B2B (Business to Business) and emergent Web Services, have served to raise the profile of on-line architectures and protocols based upon XML (Extensible markup Language) standards and its many derivative forms, such as the SOAP (Simple Object Access Protocol). For examples of implementations of Web Services see e.g. W3C (2003), Apache (2002-2004), or Sun (2002).

Indeed, the rapid pace of development in this area within industrial settings has created an equally rapid demand upon on the part of University curricula both at Undergraduate and Postgraduate levels to retain industrial credibility. We have sought to respond to this challenge by seeking to embed Web Services (more specifically XML messaging containing embedded Remote Procedure Calls (RPC's)) within our existing curriculum in "bite size" pieces.

That is to say, our aim is to expose students to a small "taster" of this leading edge and vocationally credible area, without seeking to compromise academic rigor or unbalancing their overall programme of study within our undergraduate BSc (Hons.) programmes of study.

Hence, we proceed to offer a description of a small unit of study (two weeks) embedded within an existing module together with our reflections so as to stimulate a wider pedagogic debate concerning how, and in what ways, it is possible for existing degree programmes in Computing and Computer Science to successfully seek to expose students engaged in such programmes of study to rapidly evolving industrial practice.

We describe a teaching unit based on the XML-RPC protocol (UserLand, 1998-2004)

> What is XML-RPC?
>
> It's a spec[ification] and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet.
>
> It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.
>
> *Figure 1: The definition of XML-RPC according to http://www.xmlrpc.com.*

that introduces the central techniques that all Web Services rely upon. The students require as a prerequisite a "basic" knowledge of Java. The units have been delivered in the academic years 2003-4 and 2004-5 as part of a 3rd year BSc course in "Distributed Computing" at the University of Luton. They may also be used in other contexts, e.g. in a lecture on Java programming. Each year approximately 100 students attend the class. Anecdotal evidence shows that students respond positively to the activities provided. Almost all of these students were successful in finishing their first activity. The majority of the students successfully mastered the second activity, usually with help from peers or their tutor. The core of the student experience is a balanced mixture of individual work and group work. The teaching material includes lecture slides and activities for practical sessions. It is available (free of charge) from http://perisic.com/xmlrpc (Conrad, 2003), and based on the Apache XML-RPC implementation (Apache, 2001-2003). All the code involved is open-source and can be tailored to individual situations.

The technical requirements are lightweight and minimal. At the University of Luton we use the J2SE 1.4.2 (Sun, 2004) running under a DOS shell combined with the text editor Edit+. However the activities may be performed on any Java developing environment. The computers of course need to be connected over a network. As the connection is established via port 80 (used by Web Services) there are no firewall problems to be expected.

In the remainder of the paper we firstly describe the contents and the purpose of the lecture slides. In section three that follows we describe typical activities and in section four we evaluate the material presented within the context of the learning targets addressed above. We then go on to look at the relationship between XML-RPC and SOAP. Finally we shortly discuss the student experience of activities.

2. The Lecture Slides

The first set of the lecture slides covers the following topics:

1. The role of XML in XML-RPC using the analogy with natural languages (see Figure 2 on the next page);
2. The difference between a *remote* procedure call as compared to a local procedure call;

3. A detailed discussion of the official definition of XML-RPC as presented on the homepage of XML-RPC (see Figure 1) Here the emphasis is an understanding that a distributed framework needs *both* the specification and the implementation of this specification in programming languages;
4. An extensive presentation of Java code for a Server and Client using the Apache XML-RPC implementation.

The second set contains the XML-RPC specification distributed across 13 slides with suitable annotations. This "real life" example covers addresses the following topics:

1. Example of a HTTP header;
2. The basic structure of an XML document;
3. A review of fundamental data types (int, boolean, string, double, arrays, structs);
4. Fault handling;
5. The copyright emphasising on the two main aspects that make up an open standard, i.e. that the specification must not be changed but that everyone is free to implement the specification.

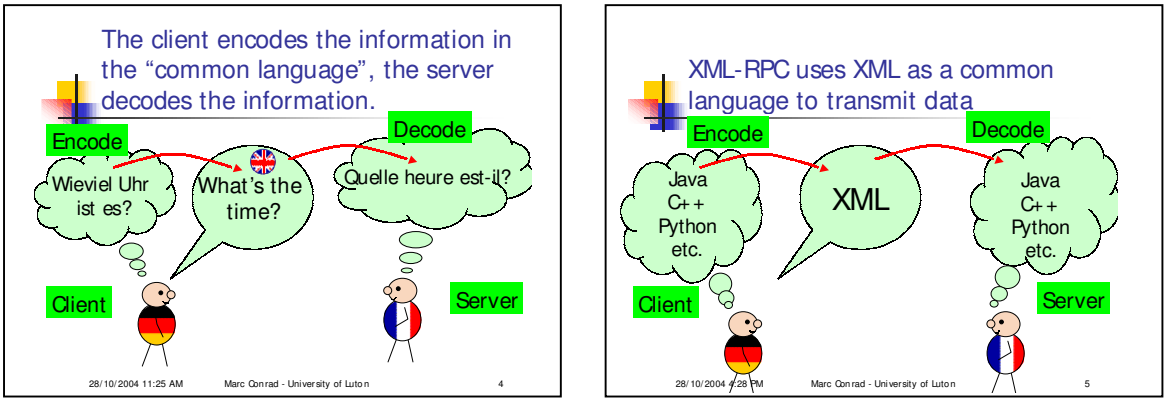

Figure 2: Using an analogy with natural languages

3. Student Activities

The first activity consists of five tasks that can be summarised as follows:

1. Compile and run a server and client program locally (the source code for these programs is provided). The basic web service is that the server on client request delivers a "Hello World" -like string.
2. Modify the code of the server such that a different string is presented (e.g. the name of the student).
3. Distribute the IP number of the machine to other students (for instance the tutor can keep a list of IP numbers with a running server on a whiteboard).
4. Change the code of the client to access a server that is not local.
5. Add a special command into the code that shows the XML that is exchanged between client and server.

Note that this activity does not require significant programming knowledge. An interesting behaviour can be observed at step 3 namely, that students who have been

successful in running their own server motivate their peers to work on their client program, therefore leading to a productive interaction.
The second activity consists of the implementation of a distributed application from the non-distributed Java program shown in Figure 3. This task can be solved using the programs of the first tasks as templates.

4. Impact on Student Understanding of Web Services
Draganova (2003) provided us with a useful template identifying the central topics in the context of web services, namely

- Discovering a location of a service provider;
- Discovering provided services;
- Providing a way of communication with a particular service;
- Providing a way to execute available functions;
- Providing a standard messaging;
- Providing a way of data representation.

```
import javax.swing.*;

public class PrimitiveChat {

// The procedure:
 public String printText(String str) {
    System.out.println( "Received: "+str);
    return "ok";
    }

 public static void main (String [] args) {
  String input;
  PrimitiveChat pc = new PrimitiveChat();

  do {
    input = JOptionPane.showInputDialog("Enter your message");
    if( input != null ) {
        pc.printText(input); // a local procedure call.
        }
    } while ( input != null );
  System.exit(0);
  }
}
```

Figure 3: A non-distributed example program

These topics can be instantiated by using XML-RPC as follows:

Discovering a location of a service provider
Students use two mechanisms to link the application: Firstly, they work on their local machine using the address http://localhost/RPC, then they access a remote location by its IP number. By giving their own IP number to peers to access their server, they are "publishing" a web service. It is then straightforward to question the usefulness of the

IP number and the possibility to replace that by a name (as given in the example in the lecture). This then naturally leads to the discussion of using nameservers etc.

Discovering provided services
In the second activity students are asked to transform a local procedure call into a remote procedure call. When they ask their peers to access this new service they quickly see that it is not only sufficient to hand over the IP number alone, but also the name of the remote procedure, therefore learning the importance of identifying the specification of a service.

Providing a way of communication with a particular service
In the last task of the first activity students examine the XML code that is transmitted between client and server. As XML is human-readable they are able to identify immediately the text inside this code. This experience of course goes hand in hand with the discussion of the XML-RPC specification in the lecture.

Providing a way to execute available functions
A remote procedure call is the simplest way to access a web service. Starting from the well-known knowledge of invoking local methods (procedures) in a Java program the activities and lectures provide a gentle generalisation of that mechanism.

Providing a standard messaging
The target of the second practical is the implementation of a "chat" system at its most basic level, therefore introducing the students to the topic of messaging.

Providing a way of data representation
One of the major advantages of using XML-RPC in the curriculum is that the XML-RPC specification can be covered *in full* in a 90 minute lecture. The data that can be transmitted in an XML-RPC request consists mostly of the usual data types encountered in a programming lecture (Strings, integers, floating points, booleans, arrays) and a very few number of data structures that may be new (structs, base64, and date).

Other topics
In addition to the primary topic of understanding Web Services, by seeing a specification as whole, students also gain a better understanding on "Open Standards".

6. XML-RPC and SOAP

The relationship between XML-RPC and SOAP is best described by Wikipedia (2005): "*[XML-RPC] was first created by Dave Winer of UserLand Software in 1995 with Microsoft. However Microsoft considered it too simple, and started adding functionality. After several rounds of this, the standard was no longer so simple, and became what is now SOAP."* Therefore XML-RPC serves as a good starting point for teaching SOAP. We especially feel that the lightweight nature of the XML-RPC protocol better facilitates the students learning experience in the given time frame of two weeks than an introduction into SOAP directly. As mentioned in section 5 two weeks are enough to cover the *full* picture of XML-RPC including examples and the specification. We see this as a major advantage as for most topics (as HTML, SOAP, XML etc.) a teaching module is able to cover only selected parts of the standard.

7. Student Experience and Feedback

The units have been delivered so far at the University of Luton twice. First in 2003 in a module of 100 students and in 2004 in a module of 160 students. In 2003 there was no formal assessment point associated to the tasks. Any feedback there is based on observation and anecdotal evidence. In 2004 the tasks have been included as part of a student portfolio that is currently (January 2005) reviewed and marked. We will report on these results at the conference and make the details available on the Internet. The experience so far shows a thoroughly positive response. We made the observation that a real environment of "working together" builds up during the assignment. While the first task of the first activity requires the student to compile and run the client and server on their local machine the following task to exchange IP numbers and to run the application distributed means that the good students usually start to help their peers in running their application (motivated by their need to find a partner where they can access a server). We see here that this activity is also valuable for enhancing "soft skills" like team working and communication.

In the second activity, running a simple chat program, the students often find it surprising that they are able to implement a service they know from their daily experience (text messaging) from scratch. A small number is often motivated to develop their software further in order to mimic features they know from other software.

8. Conclusion

By selecting out key concepts and fundamental skills (such as RPC' s) and by providing just enough industrial context, it is possible to marry vocational content with conceptual development, whilst also motivating students by relating key skills and concepts to the real world they see around them. The material presented here serves as an example on how web services can be introduced in a Java based curriculum providing the students with a positive experience on Web Services.

9. References

- Apache (2001-2003), *About Apache XML-RPC,* http://ws.apache.org/xmlrpc/
- Apache (2002-2004), *The Apache XML Project*, http://xml.apache.org/
- Conrad, Marc (2003), *Lecture slides and activities on XML-RPC*, http://perisic.com/xmlrpc.
- Draganova, Christina (2003), *Web Services and Java,* JICC 7, http://www.ics.ltsn.ac.uk/pub/jicc7/draganova2.doc.
- Sun Microsystems, Inc (2002), *Web Services Made Easier: The Java™ APIs and Architectures for XML, A Technical White Paper*, http://java.sun.com/xml/webservices.pdf
- Sun Microdystems, Inc (2004), *Java 2 Platform, Standard Edition version 1.4.2,* http://java.sun.com/j2se/1.4.2/
- UserLand Software inc (1998-2004), *The XMLRPC homepage*, http://xmlrpc.com..
- W3C (2003), *SOAP Version 1.2 Part 1: Messaging Framework*, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/
- Wikipedia, the free Encyclopaedia, *XML-RPC*, http://en.wikipedia.org/wiki/XML-RPC (accessed January 5, 2005)