Evaluation of XML-RPC interoperability between the .NET and JAVA framework

Daniel Aslan
Siemens Business Services GmbH & Co, C-LAB

Wolfgang Thronicke
Siemens Business Services GmbH & Co, C-LAB

C-LAB Report

Vol. 5 (2006) No. 1

Cooperative Computing & Communication Laboratory

ISSN 1619-7879

C-LAB Report

Herausgegeben von
Published by

Dr. Wolfgang Kern, Siemens Business Services GmbH & Co OHG

Prof. Dr. Franz-Josef Rammig, Universität Paderborn

Das C-LAB - Cooperative Computing & Communication Laboratory - leistet Forschungs- und Entwicklungsarbeiten und gewährleistet deren Transfer an den Markt. Es wurde 1985 von den Partnern Nixdorf Computer AG (nun Siemens Business Services GmbH & Co OHG) und der Universität Paderborn im Einvernehmen mit dem Land Nordrhein-Westfalen gegründet.
Die Vision, die dem C-LAB zugrunde liegt, geht davon aus, dass die gewaltigen Herausforderungen beim Übergang in die kommende Informationsgesellschaft nur durch globale Kooperation und in tiefer Verzahnung von Theorie und Praxis gelöst werden können. Im C-LAB arbeiten deshalb Mitarbeiter von Hochschule und Industrie unter einem Dach in einer gemeinsamen Organisation an gemeinsamen Projekten mit internationalen Partnern eng zusammen.

C-LAB - the Cooperative Computing & Cooperation Laboratory - works in the area of research and development and safeguards its transfer into the market. It was founded in 1985 by Nixdorf Computer AG (now Siemens Business Services GmbH & Co OHG) and the University of Paderborn under the auspices of the State of North-Rhine Westphalia.
C-LAB's vision is based on the fundamental premise that the gargantuan challenges thrown up by the transition to a future information society can only be met through global cooperation and deep interworking of theory and practice. This is why, under one roof, staff from the university and from industry cooperates closely on joint projects within a common research and development organization together with international partners. In doing so, C-LAB concentrates on those innovative subject areas in which cooperation is expected to bear particular fruit for the partners and their general well-being.

ISSN 1619-7879

C-LAB
Fürstenallee 11
33102 Paderborn

fon:	+49 5251 60 60 60
fax:	+49 5251 60 60 66
email:	info@c-lab.de
Internet:	www.c-lab.de

Contents

1. Motivation 4
1.1. Scenario 4
1.2. XML-RPC description 4
1.3. XML-RPC implementations 4
1.3.1. Apache XML-RPC 4
1.3.2. XmlRpcCS 5
2. Test setup 5
2.1. Java 5
2.2. .NET 5
3. Test results 6
4. Review of results 7
5. Listings 8
5.1. C# 8
5.1.1. Server 8
5.1.2. Client 9
5.2. JAVA 10
5.2.1. Server 10
5.2.2. Client 11
6. Acknowledgement 12

1. Motivation

Interoperability describes the ability of a system to access functions from another one. As long as the underlying operating system and the software platform are identical, components can chose between various protocols and mechanisms which usually work very efficiently. This evaluation investigates the interoperability of the two major platforms in the IT world: The JAVA platform and the .NET framework[1]. In order to generalize the results a common communication protocol has to be used, which is ideally based on XML. Despite the success of SOAP in web-like application environments, a full-fledged SOAP implementation was not desired for reasons of efficiency. Hence this evaluation uses the more basic but much more efficient XML-RPC to communicate between both platforms.

1.1. Scenario

The target platform of this investigation is a small-sized PC - the OQO[2] - which is used as one of the implementation platforms in a European research project for wearable computing. The wearable platform to be implemented actually relies on implemented components both in JAVA and .NET. In order to reduce the reimplementation and porting effort the interoperability of two co-existing framework parts is deemed necessary to satisfy some project constraints.

1.2. XML-RPC description

XML-RPC is a specification and a set of implementations allowing to make remote procedure calls (RPC) over TCP/IP connections utilizing XML as data format. Typically XML-RPC uses the HTTP protocol to communicate between two communication endpoints. The payload data of an XML-RPC call are the parameters. They can represent scalar values like numbers, strings, etc. up to complex data-structures like records or lists.

Detailed information can be found on: http://www.xmlrpc.com.

1.3. XML-RPC implementations

A number of XML-RPC implementations exist. These libraries are available for all current programming languages like PHP, Python, C/C++, JAVA, C# or Perl. For our scenario we use the XML-RPC versions for .NET, C# and JAVA.

1.3.1. Apache XML-RPC

The Apache XML-RPC implementation is very mature and offers full support for the HTTP protocol if needed. Additionally (which is interesting for local

1 .NET is a trademark of Microsoft Corp. JAVA is a trademark of Sun Microsystems Inc.

2 Http://www.oqo.com

interoperability investigations), it comes with an optimized light-weight client with only basic HTTP support. The server side of XML-RPC is not only supported by embedding XML-RPC into an existing web-server, but also comes with its own internal XML-RPC server.

Apache XML-RPC is under the Apache Software License, Version 2.0.

Link: http://ws.apache.org/xmlrpc/index.html

1.3.2. XmlRpcCS

Like Apache XML-RPC the XmlRpcCS (CS obviously stands for C#) supports both XML-RPC client and server behavior. It is not only fully XML-RPC compliant, but additionally supports the .NET Compact Framework and the open-source .NET counterpart Mono.

XmlRpcCS is under the BSD license.

Link: http://xmlrpccs.sourceforge.net/

2. Test setup

The interoperability tests took place on a machine running a .NET or JAVA client vs. a corresponding server. The development hardware as a standard desktop PC (3 Ghz, Intel P4 HT, 1 GB RAM) and the OQO – a Transmeta based shrink-sized PC with 1GHz CPU and 265 MB RAM. The operating system used was Windows XP Professional. The following environment was used for the test:

2.1. Java

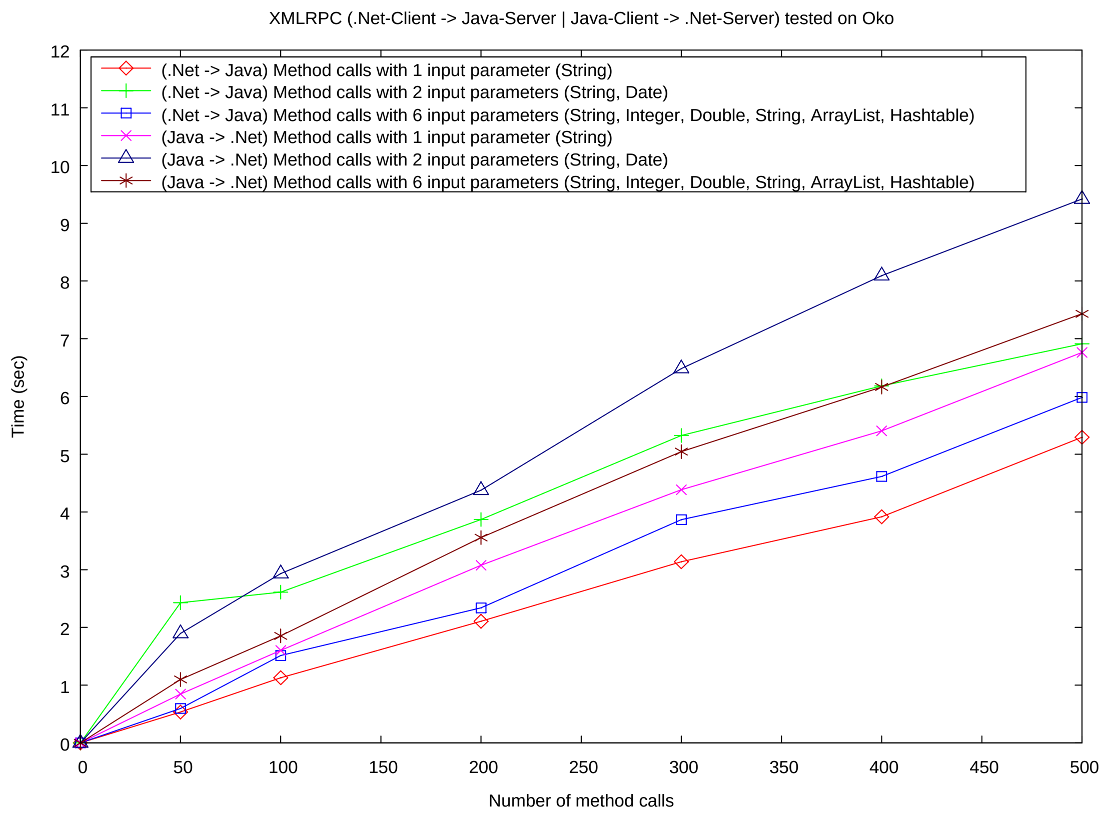
- Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_05-b04) (development PC)
- JAVA(TM) 2 Runtime Environment, Standard Edition (build 1.5.0) (OQO)
- Apache XML-RPC Version 2.0.1 + Apache Commons Codec Version 1.3 (required by XML-RPC for encode/decode XML streams)
- Used development environment: Eclipse 3.1

2.2. .NET

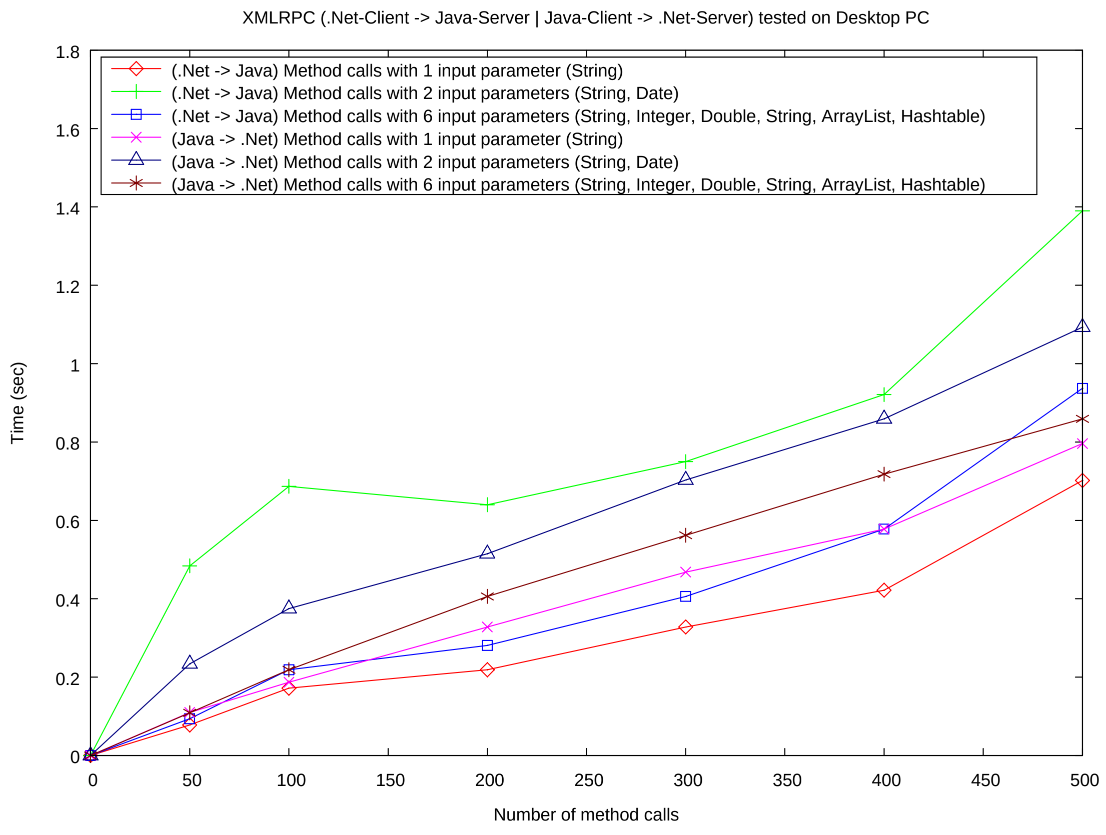
- .NET Framework SDK 2.0
- XmlRpcCS Version 1.10
- Development environment used: Visual Studio .NET 2005

3. Test results

The following figure shows the result of the test-runs on the OQO:



For comparison the tests were repeated on a standard PC:



4. Review of results

Most importantly, XML-RPC is a viable option for interoperability between .NET and JAVA. This has been proved in both calling directions. Moreover, XML-RPC is a good implementation candidate for objects of limited complexity.

The results show an approximated performance of about 50 method calls/sec on the OQO with expected variances depending on the number of arguments. Actually if a complex object is involved the construction of such objects in the test-programs causes an essential rise in computing time. The differences between server and client roles indicate differences in the implementation in the libraries used (as they are open source, they are subject to improvement). A detailed factorization was not part of this investigation.

Although “misused” in a local environment to establish a bridge between different software technologies, XML-RPC performs reliably. XML-RPC should evidently not be used, if a) continuously more than 10 messages/secs are expected (assuming that some time is needed to do some reasonable processing) or b) many different subsystems are to be coupled locally.

Aside from the local use, the strength of XML-RPC lies clearly in a distributed scenario where such calls are transferred between systems. Especially with a limited bandwidth the considerable smaller size of XML-RPC messages in comparison to SOAP messages is a great benefit.

Another benefit of using XML-RPC is that moving such components from a local communication scheme to a remote one with server/client located on different machines connected by the Internet is simple, since XML-RPC is HTTP/HTTPS driven by default.

5. Listings

The following listings present the core source of our test implementation. It has been quite straightforward and may help as a short additional guide on how to build an individual (and more functional) bridge between .NET and JAVA components.

5.1. C#

5.1.1. Server

```
/*
 * Copyright (c) 2006 C-Lab. All Rights Reserved.
 *
 * This software is the confidential and proprietary information.
 * It shall be distributed only in accordance with C-Lab.
 *
 * C-LAB MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE
 * SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
 * PURPOSE, OR NON-INFRINGEMENT. C-LAB SHALL NOT BE LIABLE FOR ANY DAMAGES
 * SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING
 * THIS SOFTWARE OR ITS DERIVATIVES.
 */
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Text;
using Nwc.XmlRpc;

namespace ObjectServer {
    class ObjectServer : XmlRpcServer {
        const int PORT = 9118;

        public ObjectServer() : base(PORT) {
        }

        public ObjectServer(int port) : base(port) {
        }

        /// <summary>Simple logging to Console.</summary>
        static public void WriteEntry(String msg, LogLevel type) {
            if (type != LogLevel.Information) { // ignore debug msgs
                Console.WriteLine("{0}: {1}", type, msg);
              }

            }


        /// Application starts here.
        public static void Main(string[] args) {
            // Use the console logger above.
            Logger.Delegate = new Logger.LoggerDelegate(WriteEntry);

            ObjectServer server = new ObjectServer(PORT);
            server.Add("container", new ObjectContainer());
            Console.WriteLine("Web Server Running on port {0} ... Press ^C to Stop...", PORT);
            server.Start();
        }
      }
}
```

5.1.2. Client

```
/*
 * Copyright (c) 2006 C-Lab. All Rights Reserved.
 *
 * This software is the confidential and proprietary information.
 * It shall be distributed only in accordance with C-Lab.
 *
 * C-LAB MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE
 * SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
 * PURPOSE, OR NON-INFRINGEMENT. C-LAB SHALL NOT BE LIABLE FOR ANY DAMAGES
 * SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING
 * THIS SOFTWARE OR ITS DERIVATIVES.
 */
using System;
using System.Collections.Generic;
using System.Collections;
using System.Text;
using Nwc.XmlRpc;

namespace ObjectClient {
    class ObjectClient {
        static string server_url = "http://127.0.01:9117";

        static void Main(string[] args) {
            int maxMethodCall = 500;
            DateTime t1;
                        DateTime t2;
            XmlRpcRequest client = new XmlRpcRequest();
            XmlRpcResponse response;

            if (args.Length > 0) {
                maxMethodCall = Int32.Parse(args[0]);
            }

            Console.WriteLine("==========Send objects - 2 input parameters - started==========");
            Console.WriteLine("Count of the method calls: " + maxMethodCall);
                        t1 = DateTime.Now;

            for (int i = 0; i < maxMethodCall; i++) {
                client.MethodName = "container.add";
                client.Params.Clear();
                client.Params.Add(i.ToString() + "_m1");
                client.Params.Add(DateTime.Now);
                response = client.Send(server_url);
                //Console.WriteLine(response.Value);
                        }

            t2 = DateTime.Now;

            Console.WriteLine("==========Send objects - 2 input parameters - finished==========");
            Console.WriteLine("Total time = " + (t2 - t1) + "msec");

            Console.WriteLine("==========Send objects - 6 input parameters - started==========");
            Console.WriteLine("Count of the method calls: " + maxMethodCall);
            t1 = DateTime.Now;

            for (int i = 0; i < maxMethodCall; i++) {
                client.MethodName = "container.add";
                client.Params.Clear();
                client.Params.Add(i.ToString() + "_m2");
                client.Params.Add(i);
                client.Params.Add((Double) i);
                client.Params.Add(i.ToString());
                ArrayList l = new ArrayList();
                l.Add(i.ToString());
                l.Add(i + 1);
                client.Params.Add(l);
                Hashtable t = new Hashtable();
                t.Add(i.ToString(), i.ToString());
                Int32 j = i + 1;
                t.Add(j.ToString(), j);
                client.Params.Add(t);
                response = client.Send(server_url);
                //Console.WriteLine(response.Value);
            }

            t2 = DateTime.Now;

            Console.WriteLine("==========Send objects - 6 input parameters - finished==========");
            Console.WriteLine("Total time = " + (t2 - t1) + "msec");


            Console.WriteLine("==========Send objects - 1 input parameter - started==========");
            Console.WriteLine("Count of the method calls: " + maxMethodCall);
            t1 = DateTime.Now;

                        for (int i = 0; i < maxMethodCall; i++) {
                client.MethodName = "container.get";
                client.Params.Clear();
                client.Params.Add(i.ToString() + "_m1");
                response = client.Send(server_url);
                //Console.WriteLine(response.Value);
            }

                        t2 = DateTime.Now;

            Console.WriteLine("==========Send objects - 1 input parameter - finished==========");
            Console.WriteLine("Total time = " + (t2 - t1) + "msec");
        }
    }
}
```

5.2. JAVA

5.2.1. Server

```
/*
 * Copyright (c) 2006 C-Lab. All Rights Reserved.
 *
 * This software is the confidential and proprietary information.
 * It shall be distributed only in accordance with C-Lab.
 *
 * C-LAB MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE
 * SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
 * PURPOSE, OR NON-INFRINGEMENT. C-LAB SHALL NOT BE LIABLE FOR ANY DAMAGES
 * SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING
 * THIS SOFTWARE OR ITS DERIVATIVES.
 */
import org.apache.xmlrpc.*;

public class ObjectServer extends WebServer {
        private static final int PORT = 9117;

        public ObjectServer() {
                super(PORT);
        }

        public ObjectServer(int port) {
                super(port);
        }

        public static void main(String args[]) {
                try {
                        //XmlRpc.setKeepAlive(true);
                        //XmlRpc.setDebug(true);
                        ObjectServer server = new ObjectServer();
                        //xmlrpcServer.acceptClient ("192.168.0.*"); // allow local access
                        //xmlrpcServer.acceptClient ("127.0.0.*"); // allow local access
                        server.addHandler("container", new ObjectContainer());
    System.out.println("Web Server Running on port " + ObjectServer.PORT + " ... Press ^C to Stop...");
                        server.start();
                } catch (Exception e) {
                        System.err.println("Could not initialize XMLRPC-Server");
                }
        }

}
```

5.2.2. Client

```
/*
 * Copyright (c) 2006 C-Lab. All Rights Reserved.
 *
 * This software is the confidential and proprietary information.
 * It shall be distributed only in accordance with C-Lab.
 *
 * C-LAB MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE
 * SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
 * PURPOSE, OR NON-INFRINGEMENT. C-LAB SHALL NOT BE LIABLE FOR ANY DAMAGES
 * SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING
 * THIS SOFTWARE OR ITS DERIVATIVES.
 */
import org.apache.xmlrpc.*;
import java.io.*;
import java.sql.Timestamp;
import java.util.Date;
import java.util.Hashtable;
import java.util.Vector;

public class ObjectClient {
        final static String server_url = "http://127.0.0.1:9118";

        public static void main(String args[]) throws XmlRpcException, IOException {
                int maxMethodCall = 500;
                Timestamp t1 = null;
                Timestamp t2 = null;
                final XmlRpcClient client = new XmlRpcClient(server_url);
                Vector params = new Vector();
                if (args.length > 0) {
                        maxMethodCall = Integer.parseInt(args[0]);
                }
                System.out.println("==========Send objects - 2 input parameters - started==========");
                System.out.println("Count of the method calls: " + maxMethodCall);
                t1 = new  Timestamp(System.currentTimeMillis());

                for (int i = 0; i < maxMethodCall; i++) {
                        params.clear();
                        params.add(String.valueOf(i) + "_m1");
                        params.add(new Date(System.currentTimeMillis()));
                        client.execute("container.addTwo", params);
                        //Object result = client.execute("container.addTwo", params);
                        //System.out.println(result);
                }
                t2 = new Timestamp(System.currentTimeMillis());

                System.out.println("==========Send objects - 2 input parameters - finished==========");
                System.out.println("Total time = " + (t2.getTime() - t1.getTime()) + "msec");
                System.out.println("==========Send objects - 6 input parameters - started==========");
                System.out.println("Count of the method calls: " + maxMethodCall);
                t1 = new  Timestamp(System.currentTimeMillis());

                for (int i = 0; i < maxMethodCall; i++) {
                        params.clear();
                        params.add(String.valueOf(i) + "_m2");
                        params.add(new Integer(i));
```

```
                    params.add(new Double(i));
                    params.add(String.valueOf(i));
                    Vector v = new Vector();
                    v.add(String.valueOf(i));
                    v.add(new Integer(i + 1));
                    params.add(v);
                    Hashtable t = new Hashtable();
                    t.put(String.valueOf(i), String.valueOf(i));
                    t.put(String.valueOf(i + 1), new Integer(i + 1));
                    params.add(t);
                    client.execute("container.addSix", params);
                    //Object result = client.execute("container.add", params);
                    //System.out.println(result);
                }
                t2 = new Timestamp(System.currentTimeMillis());
                System.out.println("==========Send objects - 6 input parameters - finished==========");
                System.out.println("Total time = " + (t2.getTime() - t1.getTime()) + "msec");
                System.out.println("==========Send objects - 1 input parameter - started==========");
                System.out.println("Count of the method calls: " + maxMethodCall);
                t1 = new  Timestamp(System.currentTimeMillis());
                for (int i = 0; i < maxMethodCall; i++) {
                    params.clear();
                    params.add(String.valueOf(i) + "_m1");
                    client.execute("container.get", params);
                    //Object result = client.execute("container.get", params);
                    //System.out.println(result);
                }
                t2 = new Timestamp(System.currentTimeMillis());

                System.out.println("==========Send objects - 1 input parameter - finished==========");
                System.out.println("Total time = " + (t2.getTime() - t1.getTime()) + "msec");
        }
}
```

6. Acknowledgement

This work was partly funded under grant 004216 by the European integrated project “wearIT@work - Empowering the mobile worker by wearable computing”.

C-LAB
Marketing
Fürstenallee 11
D-33102 Paderborn

Cooperative Computing & Communication Laboratory

Telephone	+49-5251-60-6060
Fax	+49-5251-60-6066
E-Mail	marketing@c-lab.de
URL	http://www.c-lab.de

Survey concerning your satisfaction with the report *"Evaluation of XML-RPC interoperability between the .NET and JAVA framework"*

We kindly ask you to dedicate a short moment of your time to give us your personal assessment of this report. This will help us to improve our understanding of your needs. We would like to increase the alignment of our reports with your interests to offer you a higher surplus value. Thank you for your co-operation.

How do you evaluate the subject of this report?

	applies completely			applies not at all
Topical	☐	☐	☐	☐
Interesting	☐	☐	☐	☐

How to you evaluate the content of this report?

	applies completely			applies not at all
Topical	☐	☐	☐	☐
Interesting	☐	☐	☐	☐
Articulate	☐	☐	☐	☐
Practical relevance	☐	☐	☐	☐
Informative	☐	☐	☐	☐
Innovative	☐	☐	☐	☐

Further comments:

____

____

____

____

Voluntary information:
Surname, first name: ____
Telephone: ____
E-Mail: ____

Please send the filled in form via mail, fax or e-mail to the respective address mentioned above.