WASPC SERVER IN THE CONTROL SYSTEMS
WASPC SERVER V ŘÍDICÍCH SYSTÉMECH

Abstract

This article describes implementation of the data exchanges between industrial control systems via intranet by using Web Services technology on the higher control level of the technological process. The work was realized for the assembly line of automobile components and supplemented the new functionality in the existing control system. There is industrial control system based on the SoftPLC for control the each station of the technology. The new functionality realized data exchange between each of local control system on the local stations and the master control system of the technological process.

1. The assembly line control system

The assembly line in the automobile industry includes several separate local stations and performs substep of the mounting process. The each of the local stations was operated the one’s own operator. The existing control system of the assembly link for automotive components consists the local control systems on the each assembly station and the major control system of the technology. The major control system is based on personal computer and use the operating system UNIX. The local control systems on the stations are realized on the basis of SoftPLC. They consist of the PLC (Process Logic Control) system that realizes control of the technological process and the personal computer on platform Microsoft Windows NT. It is used for the technological data acquisition, analysis and presentation. There are the main technological data signals of moment and power measurements there. The personal computer communicates with PLC system via Proibus DP industrial protocol. The local and major control systems are connected by the intranet based on the TCP/IP communication protocol. The major control system executes archiving of the measured data and further data analysis. The block diagram of the assembly link control system architecture shows the following figure 1.

Fig. 1 The assembly line control system block diagram.

Ing., Department of Robotics, VŠB-TU Ostrava, 17.listopadu, 70833 Ostrava Poruba, Czech republic, +420-602589350, viliam.gajdos@seznam.cz.

** Doc. Dr. Ing. Department of Robotics, VŠB-TU Ostrava, 17.listopadu, 70833 Ostrava Poruba, Czech republic, +420-602589350

99
There was supplemented the new functionality for the data exchange between local and major control systems in to the existing control system. To solve the data communication was used technology of the Web Services. This technology makes it possible to realize data exchange between computers on the different platforms.

2 Introduction to Web Services

Web Service is an application and information resource that produces and consumes XML documents and that can be accessed using Internet protocol (IP). The IP or transport protocols use open standards such as HTTP, HTTPS or on similar ones. Any type of application can be offered as a Web Service. Web service uses the existing IP and because it relies on XML, it is platform and language independent. Web Services are created to consume and produce messages using Client/Service communications patterns. They can also be set up to work as peers. The client then sends this as part of a SOAP message to the Web service using a standard transport protocol. The Web Service or its container deserializes the pertinent XML part of the SOAP message into data types native to the system hosting the service. Web Services essentially define messaging and messaging management mechanisms that bind together the transport and application layers. A runtime server provides a Web Services container that processes messages, dispatches them to the correct service application and implements security policies. A runtime server can operate as a stand-alone application, or can run within existing transport servers, such as IIS, iPlanet and Apache. The runtime server listens for SOAP requests from client applications. Upon receiving a request, the server then determines how to handle the message that is associated with it. It identifies which service and operation handles the message and dispatches it appropriately. Once the server determines the appropriate way to handle the message, it processes the SOAP message deserializing the XML data into system native types. The server then invokes the service. When the service completes its work, the runtime server serializes the return value into XML, packages it into a SOAP response message and sends the message back to the calling application.

2.2 Simple Object Access Protocol (SOAP)

SOAP is an XML based markup language. The core protocol underlying Web Service to defines a standard message format for carrying data objects. Typically, when using SOAP in a Remote Procedure Call (RPC) context, a client generates a request message that it sends to the service. The server then generates the response message. Depending on the rules of how its contents should be serialized, the body of a SOAP document contains one or more objects to be consumed by the receiving application. The root of a SOAP message is an envelope in which a developer can place the XML representation of these objects. The SOAP envelope can also contain routing, state and security information. These are placed in one or more headers. Web Services can handle SOAP messages in two different ways. In their simplest form they wrap function arguments and return values. The second approach is to treat a SOAP message as a one-way document containing information to be handled by a service with the response message optional.

2.3 Web Services Description Language (WSDL)

WSDL is an XML based markup language used to describe and define the Web Services. The Web Services Description Language (WSDL) is used to describe Web Service interfaces for connecting providers and requesters. This means methods as well as implementation endpoint URLs and other transport details. The WSDL document makes public your Web Services. Providers register information about their Web Services in the UDDI directory and query it to find Web Services that meet their needs. It describes what the Web Service does, how it communicates, and where it resides. UDDI operations can be invoked programmatically using a SOAP client or manually via a user interface. A client application developer uses the WSDL document at development time to generate data types to be placed in SOAP messages and service interface stubs which are then compiled into the application.
2.4 Universal Description Discovery and Integration (UDDI)

UDDI provides a standard mechanism for publishing and discovering Web Service descriptions. It provides the meta-information needed to decide which service an application should use and where it can be found. Clients can query a UDDI registry based on company name, industry category, taxonomy type or other criteria. UDDI then provides pointers to WSDL documents that describe services and one or more implementations a client might choose to access. Seeing as service locations and implementations may change over time due to issues like load balancing, maintenance and versioning, UDDI is an excellent way to indirectly manage services.

2.5 Extensible Markup Language (XML)

Extensible Markup Language (XML) defines a standard structure for documents that contain structured data. At a broader level it is also a standard for developing structured languages for data interchange in essence a metalanguage. XML is not a computer language in the usual sense, but a standard for creating data interchange documents and languages. XML is a text based human interpretable way to represent data and data structures using a subset of the Structured General Markup Language (SGML). Web Services use XML to describe application class interfaces and to encode messages. Any number of libraries and APIs made available in recent years can deserialize XML data into native data types and can serialize native data types into XML. Web Services use these libraries to wrap native applications or to build XML cooking and munching applications from scratch. They make applications accessible to clients and peers written in any language, running on any platform as long as a reliable XML handling library can be provided or found to do the serialization dirty work. The difference between XML and HTML is that definition XML is extensible. Developers can define extended documents using appropriate data node and attribute names. The structure of these documents can be declared with other documents. This is done with a Document Type Definition (DTD) file or to define document structure using XML schema.

2.6 Introduction to WASP Server for C++

WASP Server for C++ (WASPC) is a standards compliant, development environment for creating, consuming and securing Web Service messages and documents in C/C++ based applications. It is an ideal solution for exposing the functionality of existing C/C++ applications as Web services, and can be used to easily bridge between Java, C++ and MS .NET applications running on different platforms. WASPC offers unique portability, modularity and a comprehensive API that allows it to be seamlessly ported into existing applications. WASPC also provides the tools necessary for designing, coding, testing and deploying the Web services. The API offers all the classes and macros needed for data serialization, binding to transports, securing connections, wrapping existing code into a web service, or adding a web service to tested applications. WASPC is guaranteed to interoperate with a broad range of SOAP implementations, including Microsoft .NET, J2EE servers, Apache AXIS, and others. The architecture of WASPC shows the following figure 2.

![WASPC server architecture](image)

Fig. 2 WASPC server architecture.

101
It also supports industry standard security protocols. Included utilities can be used to generate skeleton and stub code, application WSDLs, and security information. It offers unique portability, security, modularity and the comprehensive API libraries.

3 Integration Web Services to the existing control application

The Web Services technology was used for realize communication between control systems on the assembly link. The current control application was done in C++ programming language. Therefore it was necessary to realize the new functionality of the data exchange in C++ programming language. The control system was operated data acquisition, analysis and data presentation for operators. At the same time the system communicated with industrial control system based on PLC and took part in control of the technologic process. The system calculated a lot of parameters of the measuring data signals (torque and force). The control system of the each station on the assembly line sent some of the parameters to the master control system after finish of the each measurement. The master control system executed archiving and next basic analysis of the parameters values.

3.1 The Web Services Server

For development and compilation WASPC server on UNIX platform is necessary to use compiler gcc version 3.3.3 or compatible. For compilation was currently used the compiler gcc version 3.3.2. The Web Service loosely corresponds to a C++ class and an operation corresponds to a method. All the Web Services was created as classes with methods and implemented on the server. It only remains to take this service and integrate it with the WASPC runtime libraries. It was necessary to link the service with the WASPC server code in order to create a SOAP enabled executable. The services were compiled and deployed as a shared library. The first a class factory was created that can instantiate an instance of the service. WASPC will use this factory to create the service object when needed. Next, this factory was added to an array of such factories. The array of factories is then registered with the WASPC super factory. Finally, the WASPC server infrastructure was initialized and the runtime was started. The runtime started up on default port number 6070 as an http listener and started accepting requests. It will block forever until killed or politely shutdown. The following code example betokens source code of the server implementation.

```cpp
#include <iostream>
#include <wasp/common.h>
#include <wasp/runtime/Runtime.h>
#include <wasp/runtime/SuperFactory.h>
#include "ControlService.h"

WASP_FACTORY_DEFINE (ControlService); // The factory array

int main (int, char **) {
    WASP_FactoryDefinition serviceFactory[] = {
        WASP_FACTORY_ENTRY (ControlService),
        WASP_FACTORY_END ()
    }

    WASP_Runtime::serverInitialize();
    WASP_SuperFactory::registerFactory (serviceFactory);

    try {
        cout << "Starting WASP Server on port 6070" << endl;
        WASP_Runtime::serverStart("config.xml", NULL);
    } catch (WASP_Exception *exc) {
        delete exc;
        WASP_Runtime::serverTerminate();
        cout << "WASP Server shutdown" << endl;
        return 0;
    }
}
```
The code first includes three mandatory WASPC header files. It also includes the header file for the current service. After that begins the registration process. At the beginning it uses the WASP_FACTORY_DEFINE macro to create the class factory for the service. In the body of main() function it populates the array of factories. Each array element in the serviceFactory array holds a struct containing the name of the service and a pointer to the factory function that instantiates it. WASPC initializes all of the server infrastructure with the serverInitialize() call and registers the array of service factories with what’s called the WASPC factory via the registerFactory() call. The configuration file config.xml contains the namespace declarations and server configuration elements. There are a lot of components in the attribute list. The important ones are the wsdl and URL attributes. The wsdl attribute describes the path to the service’s WSDL document relative to the location of the configuration file and the url attribute specifies the URL of the service relative to the root of the HTTP server. Finally, serverStart() function is called. This is what tells WASPC server to start listening for and handling requests.

### 3.2 The Web Services Client

There was added current path for the current environment WASPC home directory in the Microsoft Windows operating system variable named PATH:

```
D:\temp\wasp\wasp-cpp-5.0-win-vc70-bin\bin
```

The C/C++ programming language environment preprocessor was configured like:

```
_LIB="WASP_DEBUG;WIN32.
```

The new functions and mandatory header files for the WASPC client implementation were added into the application source code. These functions call the Web Services on the server.

```c
#include <wasp/common.h>
#include <wasp/runtime/Runtime.h>
#include <wasp/rtt/ioperators.h>
#include "hello.h"
```

The first step in the local function is to client initialize from the initialization file in the XML format. Implementation of the calling current Web Service is effected by the standard WASPC method. On the server side was initialized the appropriate service. The following code example betokens calling of the Web Service for send the current data package to the server.

```c
FUNCTION_CONOECTION( WASP_CLIENT )
{
   
   try {
      
      // Initialize WASP_client.
      WASP_Runtime::clientInitialize();

      // Start client using the specified XML configuration file.
      // This default client configuration file is part of the WASPC++ dis-
      // tribution.
      WASP_Runtime::clientStart("conf/client.xml",NULL);
      SendValuesService _sendService;

      // Call the service to send current values into server.
      _sendService.sendValues(_MaxValue, _MinValue);
   } catch(WASP_Exception *exc) {
      
      WASP_Runtime::clientTerminate();
   }

   // Terminate client
   WASP_Runtime::clientTerminate();
}
```
4 Conclusion

The article illustrates how to use the Web Services communication technology for sharing data between a few industrial control systems on the different platforms. This practical application shows advantages using of this communication technology at solve of more complicated data transfer tasks. The Web Services technology is opened system configurable according to the concrete application requirements. They are applicable in all lines of industry, in a various technologies and techniques. The importance and utilization of the Web Services will be greater in future, mainly in complicated control applications.

References


Opponent: Ing. Petr Orság, Ph.D.