Distributed Automation System based on Java and Web Services

Nikolay Kakanakov, Mitko Shopov, Grisha Spasov

Abstract: *The paper presents the implementation of a model for Distributed Automation Systems which is experimentally built in the laboratory for Distributed Systems and Computer Networks (http://net-lab.tu-plovdiv.bg/). It discusses the N-tier model and its integration to the filed of distributed automation. The implementation of service-oriented middleware for interaction between tiers in the model is proposed. The system is based on enterprise web portal technology for the realization of presentation tier and web services for interconnection between middle tiers. The implementation is flexible and scalable, by means of using open and popular technologies for enterprise application development and integration. The main aspect of presented system is spreading the work of applications for distributed automation over large distances. In the paper the distribution of automation services – lookup and registration (dynamic UDDI) and program-to-program interaction (SOAP) are discussed. The implemented system can be applied to Distributed Automation Systems and makes the integration of automation and business logic of an enterprise feasible.*

Key words: *Distributed Automation, N-tier Models, SOA, Web Services.*

INTRODUCTION

Design and development of Distributed Automation Systems (DAS) goes in a new direction over the past few years – towards standardization, openness and integration with other business entities. High-level programming languages, component-based platforms, Internet technology, and standardized communication interfaces, all influence the development of today's DAS [4, 5].

Following this trend of progress, adaptation of enterprise application architecture to the field of automation is now feasible. Many organizations work on developing middleware technologies for Distributed Automation. An interesting solution using the latest technologies is the Web Services Architecture (WSA) – so called "middleware for middleware". It is believed that WSA has the potential to spread automation systems over Wide Area Networks [4, 7, 9, 10].

The presented paper discusses a model for integration of enterprise technology and distributed automation. The implementation of service-oriented middleware for interaction between tiers in N-tier architecture for DAS is proposed in the paper [8].

Multi-tier approach

Multi-tier architectures provide many benefits over the traditional client/server architectures [3, 10, 11]:

- Installing and deploying the user interface is virtually instantaneous - only the Web interface in the middle tier needs to be updated.
- Without a "thick" client interface, it is easier to deploy, maintain, and modify applications - no matter where the client is located.
- Because the application itself is server-based, users always access the most up-to-date version.

These benefits explain the growing popularity of the multi-tier architecture, and why almost every client/server application provider has retooled or is retooling to support Web-based clients [3, 10, 11].

The integration between tiers is by means of a middleware technology. All middleware technologies have the same problems and aims – high performance, flexibility, interoperability, scalability and application-to-application interaction. The most promising middleware nowadays is WSA. Its popularity derives from its basic features [3, 10]:

- Ubiquitous infrastructure – they operate over standard TCP/IP networks and use ubiquitous HTTP/SMTP for transport.

- Proven approaches – they use both message-oriented and RPC-based interaction which makes them flexible.
- XML – they do not need specific IDL for describing the interfaces and the data entities are self-describing.
- Business standards – Business-to-Business interaction is by means of standard documents and processes.

The most important strength of WSA is that it accommodates diversity and heterogeneity, not only in platforms, but also in middleware systems.

Enterprise portal technology

Spreading of enterprise resources over large distances, together with complexity and expensiveness of modern software, has motivated many companies to invest in enterprise portals as a mechanism by which they can manage information in a cohesive and structured fashion. Among the benefits of using enterprise portals are: they provide a single point of entry for different type of clients; portals can access Web services transparently from any device in virtually any location; they provide the ability to integrate disparate systems and leverage the functionality provided by those systems. Portals have so many advantages, that they have become a standard for Web application delivery [14].

APPLICATION OF N-TIER MODEL FOR DEVELOPING DISTRIBUTED AUTOMATION SYSTEMS USING JAVA AND WEB SERVICES

The implemented system is based on N-tier model for distributed automation [8]. This model generally consists of four tiers which are separated in their functionality and administration (figure 1).

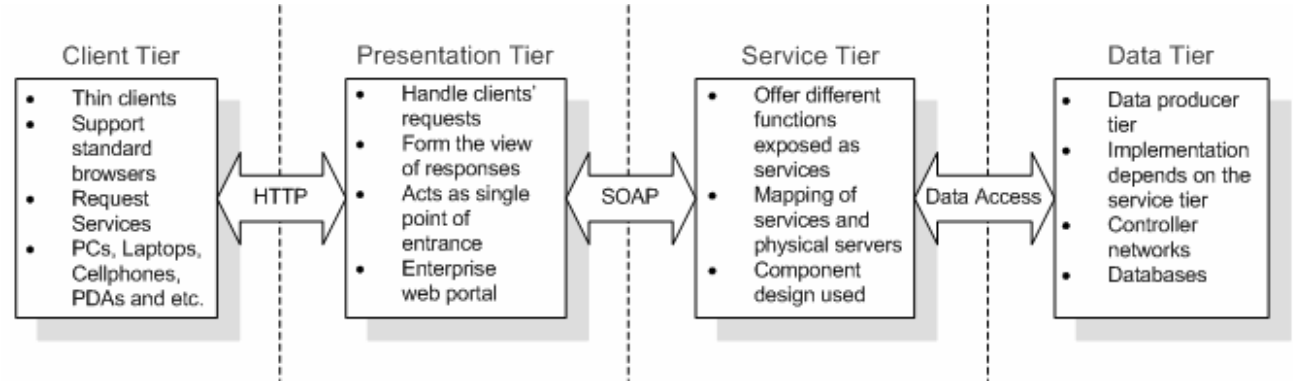

Figure 1: N-tier model for distributed automation.

The functions of the tiers are as follows:

- *Client tier* – It works on the top of the model. The clients request services from the system using regular Internet browser or via web services. Different responses can be constructed depending on the client's platform (PC, PDA, cellphone).
- *Presentation tier* – It is responsible for handling the requests and forming the responses. The requests are analyzed and dispatched to the appropriate service on the next tier. The server working on this tier is called the enterprise Web Portal and is based on the portal technology.
- *Services tier* – On this tier the main functionality of the model is placed. The different services work on different servers, so failure of a single server affects only the corresponding service. This modular approach increases the flexibility and reliability of the system.
- *Data tier* – the role of the data tier is to produce and/or store data. It depends on the corresponding upper tier server. In case of data logging service it can be a Database. In case of automation services – a controller network.

The format of the messages exchanged between individual tiers is chosen for best performance, universality, and scalability. Clients interact with the Portal via HTTP – a

HTML documents or a SOAP message. The service tier and the data tier are interconnected via standard (JDBC, ODBC) or custom protocols (CNDEP - Controller Network Data Extracting Protocol [12]). The Presentation and the Service tiers communicate using web services, as it is described in the following section.

Service Oriented Middleware for DAS

The interconnection between the Presentation and the Service tier is the backbone of the system – its middleware. The middleware technology chosen is based on web services because of their ability for application-to-application interaction. The portal first locates the web services provided by the service tier and then uses them in proper order and combination to achieve the result issued by the client.

Every server from the Service tier provides one or more web services according to its functions and publishes them in central directory. The type and number of the provided services for every server depends on its role. It can provide common services like: logging, accounting, grouping and/or structuring the results of other services; or real automation services like fetching data from sensors or sending commands to actuators, working on the Data tier. In this case provided services are based on functions of controllers in corresponding controller network.

A similar Remote Service Architecture is proposed by N. Jazdi in [5]. The main idea of the paper is to present the functions of embedded devices as services on a gateway server.

Localization Services

As documented in [6], there are two general ways for localization of remote services. The first one is to build up a central directory, in which any relevant information is stored. The information can be directly requested from the directory by a client. The other possibility is to carry out a Peer-to-Peer (P2P) search, that is to say, you prepare a current image of resources by a live search.

Usually, P2P networks do not have any fixed topology and as a result are self-organized. The basic idea behind them is that every peer knows its neighbor and consequently the neighbor's neighbor and so on. Therefore, a failure of one peer would not cause a failure of the whole network [6].

In the central directory approach, a separate server is used to store information about available services. Clients can retrieve information from the server at any time. Therefore, the service provider has to register its information to the server, before it can be used by the clients. Popular implementations of central directory are Jini, Universal Plug and Play (UPnP), and Universal Description, Discovery, and Implementation (UDDI) [1, 6].

In the current paper, a central directory approach – UDDI, is chosen, as long as it is a part of the WSA specification. It defines how to interact with a registry and the format of the entries in it. Interactions with the registry are of two types: registration and lookup. There are two types of UDDI registries: public and private. Public ones are accessible to everyone and play the role of open search engines for Web services. Private ones are those that enterprises create for their private use. For obvious reasons, industrial strength Web service implementations are to be based on private repositories. The use of dynamic binding between client application and service factory is a double edge sword. If the dynamic binding is used simply to determine the location of a well defined service, it is indeed a useful feature. Any other form of dynamic binding will make it almost impossible to develop real applications [1].

Sample Implementation

In this section a sample implementation is presented. The architecture of the system is shown on figure 2. It consists of a UDDI register, transaction servers (TS) and a web portal, repositioned on separate machines. The transaction servers are deployed on Apache Tomcat 5.5.14 and for the UDDI register a Microsoft UDDI Services is used. There

are two measurement services and a calculation service deployed (see figure 4). All of the services are accessed from the web portal. The calculation service also access measurement services to collect data and perform a function over them (*average* in current implementation). A well defined web services are used and the UDDI register is used only to determine their location. Data producer components for the measurement services are controller networks (figure 2).

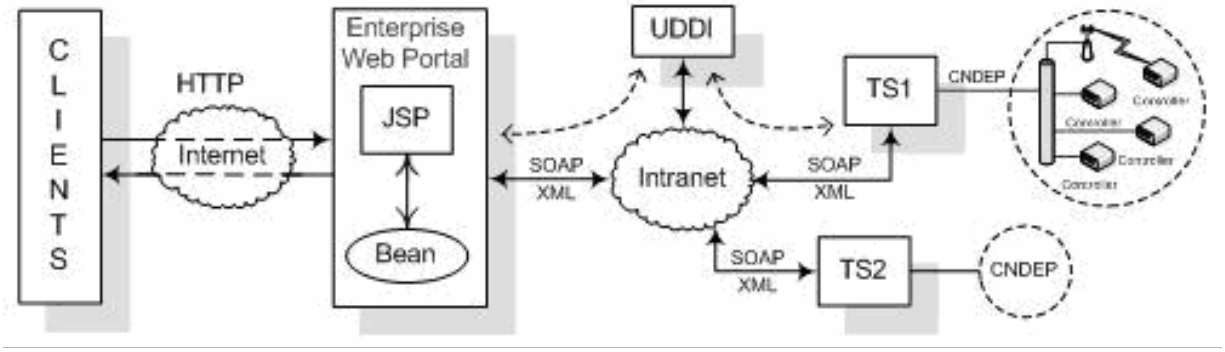

Figure 2: The architecture of implemented system.

The web portal provides a single point of entry for clients. It is built up from the following components (figure 2): presentation component – a Java Server Page (JSP) and a binding component – a Java Bean, used to transparently locate and call the right web service. A fragment from the JSP page that figures out the integration of these two components is shown on figure 3.

```
<JSP:useBean id="ts" class="WSClients.TsClient" />
Average Temperature: <JSP:getProperty name="ts" property="temperature" />
Average Humidity:  <JSP:getProperty name="ts" property="humidity" />
Location: <JSP:getProperty name="ts" property="locations"/>
```

Figure 3: Invocation of services in JSP.

A fragment from a WSDL file describing the interface of a temperature measurement service is shown on figure 4. The format of request and response messages can be seen.

```
<!--WSDL created by Apache Axis version: 1.3 -->
<wsdl:message name="getTemperatureRequest">
  <wsdl:part name="in0" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getTemperatureResponse">
  <wsdl:part name="getTemperatureReturn" type="soapenc:float"/>
</wsdl:message>
```

Figure 4: Service interface description from WSDL.

The data entities trasfered between the portal and transaction server and the service functions are enveloped in SOAP. The SOAP transport chosen is HTTP. This allows distribution over networks separated by firewalls. The encapsulation of SOAP message in HTTP body is exposed on figure 5a (Request) and figure 5b (Response).

The Request consists of addressing the *getTemperature* web service and calling its function *"Average"*. This function contacts all registered measurment services for a particular location and returns the arithmetic mean of their responses (assuming the responses are float numbers representing temperature). The value data in the Response is not XML encoded because it is of a simple data type – float.

```
POST /axis/services/TemperatureMeasurement HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.3
SOAPAction: ""
Content-Length: 506

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
   <ns1:getTemperature soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:ns1="http://DefaultNamespace">
     <in0 xsi:type="soapenc:string" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
       Average
     </in0>
     </ns1:getTemperature>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 5a: SOAP request enveloped in HTTP POST.

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/xml;charset=utf-8
Date: Mon, 10 Apr 2006 09:16:54 GMT
Connection: close

<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
   <ns1:getTemperatureResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:ns1="http://DefaultNamespace">
   <getTemperatureReturn xsi:type="soapenc:float"
      xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
       20.445
   </getTemperatureReturn>
   </ns1:getTemperatureResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 5b: SOAP reply enveloped in HTTP.

CONCLUSIONS AND FUTURE WORK

The implemented system is based on multi-tier architecture which makes it very flexible and scalable. The tiers are functionally separated for increasing security and reliability. Administration or maintenance of a server on a particular tier did not affect the other tiers. Component approach in designing allows interoperability and reusability.

The proposed interconnection middleware between Presentation and Service tiers is service-oriented. This allows the system to spread over wide area networks (by means of VPNs). In that way the service tier is distributed over large distances, which is applicable for corporate automation businesses.

Popular and standard technologies are used for implementation of every tier: on the Client tier is Web browser; on the Presentation tier is the portal technology; on the Service tier – web services. Only on the Data tier, where the real automation takes place, there are custom protocols for communication with controllers.

The paper presents the implementation of a model for Distributed Automation Systems which is experimentally built in the laboratory for Distributed Systems and Computer Networks [13]. The future work includes the experimental analysis of the system, evaluation of request/response times, estimation of the effectiveness in a function of the server or network load. The other direction for evolution of the model is applying web services architecture to the Data tier – directly to the embedded devices. [2, 4].

ACKNOWLEDGEMENTS

The presented work is supported by National Science Fund of Bulgaria project – **"ВУ-966/2005**", entitled "Web Services and Data Integration in Distributed Automation and Information Systems in Internet Environment", under the contract "**ВУ-МИ-108/2005".**

REFERENCES

[1] Alonso, G., Myths around Web Services. IEEE Data Engineering Bulletin, Volume 25, number 4, 2002.

[2] Engelen, R. van, Code Generation Techniques for Developing Light-weight XML Web Services for Embedded Devices, ACM SAC'04, March 14-17, 2004, Nicosia, Cyprus, pp. 854-861, ISBN:1-58113-812-1.

[3] Fowler, M., Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 1st Ed., 5 Nov. 2002, ISBN: 0-321-12742-0.

[4] Jammes, F., H. Smit. Service-Oriented Paradigms in Industrial Automation, Industrial Informatics, IEEE Transactions on Volume 1, Issue 1, Feb. 2005 pp. 62 – 70.

[5] Jazdi, N., Component-based and Distributed Web Application for Embedded Systems, International Conference on Intelligent Agents, Web Technology and Internet Commerce, 9-11 July 2001, Las Vegas, USA.

[6] Jazdi, N., J. Konnertz. Localization of distributed internet ready automation devices. International Conference on Intelligent Agents, Web Technology and Internet Commerce, 12-14 February 2003, Vienna, Austria.

[7] Kakanakov, N., G. Spasov, Adaptation of Web service architecture in distributed embedded systems, Proceedings on the International Conference – CompSysTech'05, pp. IIIB.10-1 – IIIB.10-6, 16-17 June 2005.

[8] Kakanakov, N., M. Shopov, G. Spasov, A New Web-based Multi-tier Model for Distributed Automation Systems, Journal "Information Technology and Control", Year IV, 2006 (in press).

[9] Topp, U., P. Müller. Web based service for embedded devices. Lecture Notes in Computer Science, Volume 2593 / 2003, pp. 141 – 153, ISSN: 0302-9743.

[10] Vinoski, S. "Where is Middleware?" *IEEE Internet Computing,* March/April 2002, vol. 6, no. 2, pp. 83-85.

[11] Youngblood, G. M., Smart Environments, Ch. 5: "Middleware", pp. 101-127, 2004., ISBN: 0-471-54448-5.

[12] http://net-lab.tu-plovdiv.bg/CNDEP/ - Controller Network Data Extracting protocol.

[13] http://net-lab.tu-plovdiv.bg/ - Virtual Laboratory of Computer Networks and Distributed Systems.

[14] http://portals.apache.org/ - The homepage of the Apache portals project.

ABOUT THE AUTHORS

Nikolay Kakanakov, PhD Student, Department of Computer Systems and Technologies, Technical University of Sofia, branch Plovdiv, Phone: +358 32 659 758, e-mail: kakanak@tu-plovdiv.bg.

Mitko Shopov, BSc graduate student, Department of Computer Systems and Technologies, Technical University of Sofia, branch Plovdiv, Phone: +358 32 659 758, e-mail: mshopov@tu-plovdiv.bg.

Assoc. Prof. Grisha Spasov, PhD, Department of Computer Systems and Technologies, Technical University of Sofia, branch Plovdiv, Phone: +358 32 659 724, e-mail: gvs@tu-plovdiv.bg.