A New Web-based Multi-tier Model for Distributed Automation Systems

N. Kakanakov, M. Shopov, G. Spasov

Keywords: Multi-tier model, Distributed Automation systems, J2EE, XML

Abstract: In this paper a new Web-based multi-tier model for Distributed Automation Systems is proposed. The presented model is constructed for scalability, flexibility and platform independence. Based on the model, integration of automation and business information systems is attained. The model consists of four tiers: Client tier, Presentation tier, Services tier and Data tier. To achieve reliability and security a functional separation of the services' roles is suggested.

The clients' interaction with the system is based on standard HTTP. The Presentation tier is implemented using J2EE technology and acts as point of presence for the services provided by the Services tier. The main application logic of the model is located on the Services tier. The communication between Presentation and Services tiers is based on XML encoded messages. The Data tier incorporates Databases and Controller Networks.

1. Introduction

Computer technology and automation is currently built on a foundation that assumes the existence of a perfect and complex infrastructure. That is why even a single failure can ruin the whole systems. Among these complex systems are Distributed Automation Systems (DAS), composed of many different nodes in need to communicate with each other. The tasks for these systems require reliable and predictable performance even though the nodes are individually unreliable. This approach is now feasible because new technologies are present for increasing scale of integration, and because on the market today there are a plenty of new automation devices with integrated TCP/IP stack and Web server [1, 10].

The similarity of automation and administrative systems originates the idea of integrating these two sides of a business. To accomplish this, the implementation should rely on standard, flexible and innovative model. This stimulates adaptation of well-known business standards in DAS [5, 7, 10].

The presented multi-tier client/server model is based on standard transport protocols, universal data description language (XML) and web-enabled microcontrollers. Its design is oriented towards achievement of platform independence, flexibility and scalability.

2. Related work

Fortino, et al. [3] and Pianegiani, et al. [6] have proposed systems for distributed measurements based on abstract three-tier architecture. In these systems a Java Applet is used as client interface, Java-based server presents the middle tier and the connection with the measurement devices is based on RMI and traditional sockets.

A model for distributed measurement of temperature and humidity has been proposed by Spasov, et al. [7]. The paper focuses on adaptation of standard business model in distributed measurement. On the data tier the database is substituted by controller network.

Jazdi in [4] has proposed a model for adapting Web technologies in Industrial Automation. In this model the functions of embedded devices are presented as services on the middle-tier server, called "Remote Service Server".

A method for remote management of gas chromatograph is proposed by Topp et al. [8]. The paper presents two different approaches for realization of remote management – CGI-based and SOAP-based.

Winiecki et al. [9] have proposed a Java-based software environment for designing virtual measuring instruments. The presented specialized server, with a controls library, allows distributed implementation of a measuring system.

3. Background

Multi-tier architectures are traditionally used for database applications. The middle tier separates presentation and business functions and its services allow communication between programs based on different technologies and programming languages [10].

Different technologies for realization of the middle tier exist (e.g. transaction processing, message-oriented, object-oriented, and Web-based). They differ in communication protocols and service allocation [10].

Multi-tier architecture provides many benefits over traditional client/server architecture [12]:

- 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 [12].

4. Web-based Multi-tier model.

The presented model generally consists of four tiers. The tiered architecture is chosen for flexibility and separation of presentation and business roles within the real automation. The additive benefit is the security – every tier can connect only to its direct neighbors. So the data and the business rules cannot be directly accessed from the Internet, and thus cannot be harmed (see figure 1).

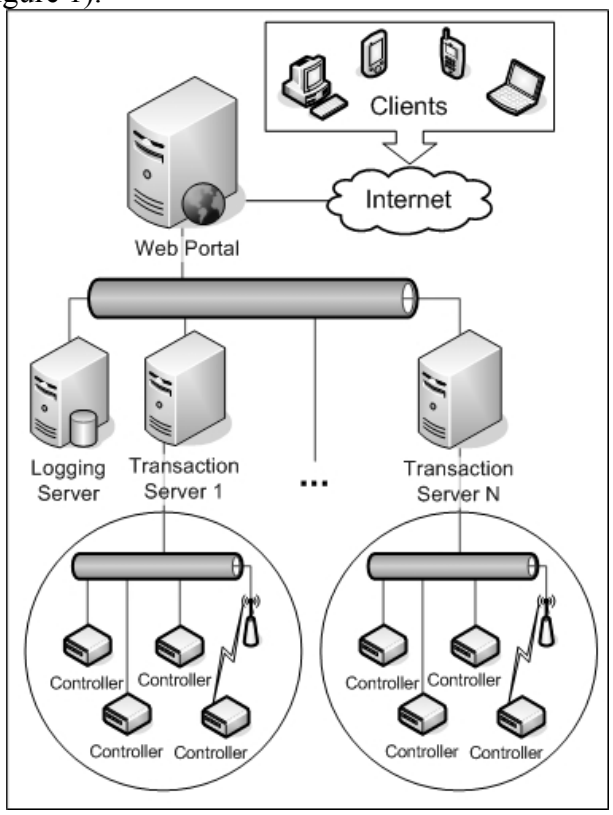

Figure 1: Architecture of the presented model.

On the top tier are the clients – ***Client tier***. They request services from the system using standard internet browser. Different HTTP pages are constructed for the different kinds of devices (e.g. PCs, Laptop, PDA, cellphone).

The next tier is the ***Presentation tier***. This tier is responsible for handling the clients' requests and forming the view of the responses. After receiving a request it is analyzed, transformed into XML encoded queries, and dispatched to the appropriate server from the next tier.

The ***Services tier*** works bellow the presentation tier. On this tier all the functionality of the model is placed. The different services work on different servers, so a hardware failure of a single server affects only the corresponding service. This modular approach increases the flexibility and reliability of the whole model. Example functions of the servers in the services tier are: data logging, plant control and monitoring, commerce services, customers support services, training services, accounting services, etc (figure 1).

The lowest tier of the model is the ***Data tier***. Its role is inherited from the three-tier database model. It depends on the upper tier servers. In the case of logging server the data tier is a database. In the case of transaction server the data tier is presented of controller network (figure 1). Other forms of data tier occur in other cases. Generally the role of the data tier is to produce or store data.

5. Functional Description of the tiers

Next, a more in-depth description of the tiers, their structure, functions, and interfaces is given. The format of the messages exchanged between individual tiers is chosen for best performance, universality, and scalability. Client and Presentation tiers interact through HTML/HTTP. Communication between Presentation tier and Services tier is based on XML, and the last two tiers communicate through standard (JDBC, ODBC) or custom protocols (CNDEP - Controller Network Data Extracting Protocol [11]) (figure 2).

5.1. Client tier.

On this tier, the clients of the system reside. A client can be any device with a standard internet browser (PC, Laptop, PDA, Cellphone) (figure 2a). Because the communication with the Presentation tier is based on exchange of standard HTTP request/response pairs, there is no need for extra plug-ins, keeping clients as thin, as possible.

Clients request services by using HTTP POST method. There are two possibilities for service request – device-oriented or data-oriented. If the individual device that offers the service is known, the device-oriented request is used. If the device that offers the service is unknown or combined request to two or more devices is required – the data-oriented request is used. The response is a simple hypertext document, specific for different types of clients.

5.2. Presentation tier.

This tier is represented by a Web Portal (figure 2b). It is a web server, which acts as point

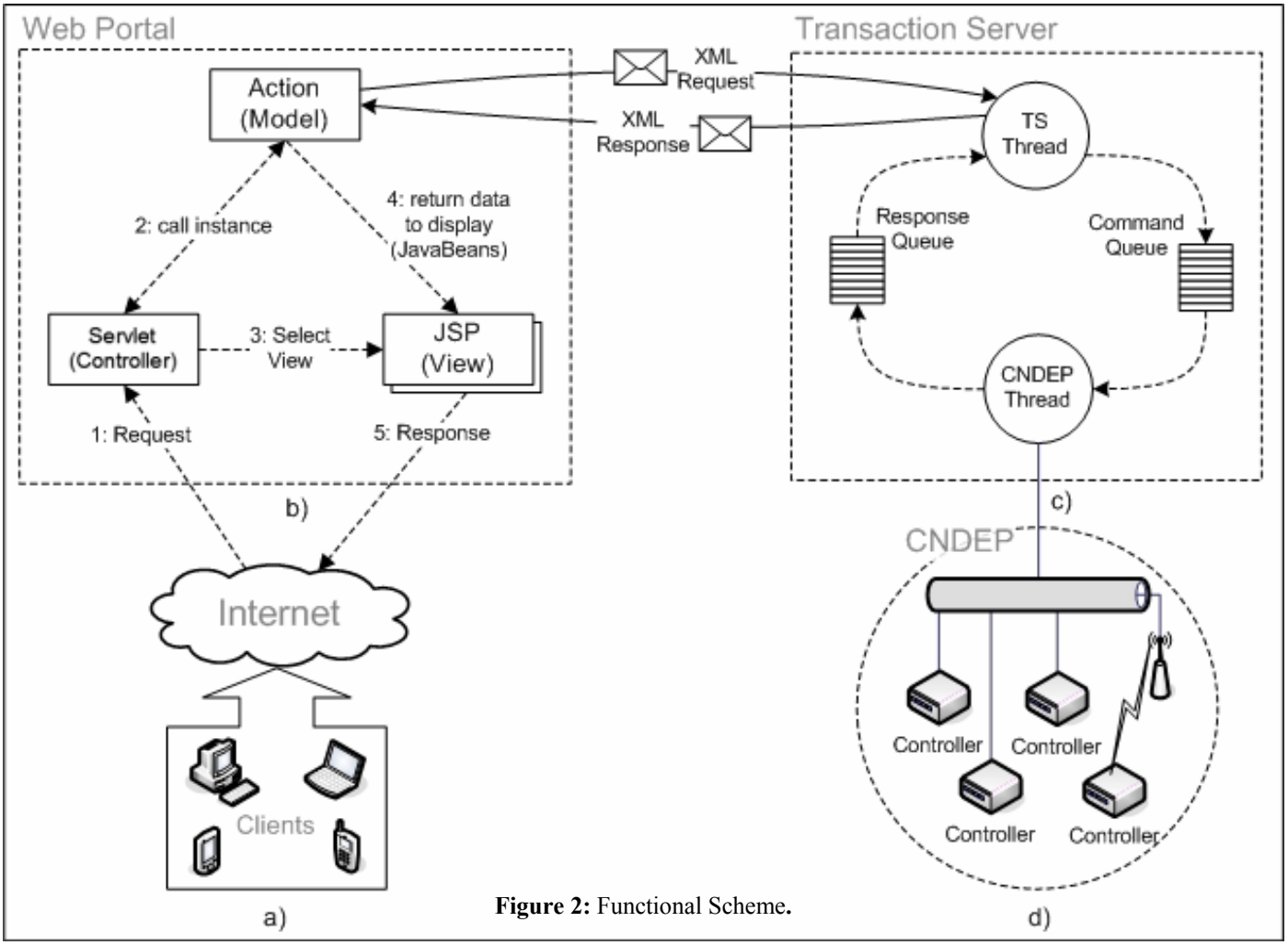

Figure 2: Functional Scheme.

of presence for different services, offered by the system. The popular Model-View-Controller architecture [2] is used. Its implementation is based on Sun Microsystems JavaServer Pages™ technology [12].

```
<?xml version="1.0" encoding="UTF-8"?>
<cndep:Request xmlns:cndep=
  "http://net-lab.tu-plovdiv.bg/das"
  xsi:schemaLocation=
  "http://net-lab.tu-plovdiv.bg/das DAS.xsd">
  <target name="IPC@CHIP">
    <Command type="Get">
      <name>Temperature</name>
      <id>0</id>
    </Command>
  </target>
  <Query>GET temperature FROM *</Query>
</cndep:Request>
```

Figure 3: XML Request.

Controller functions are handled by a Servlet (figure 2b). It processes all the HTTP requests and forms XML queries. The Servlet offers services for user authentication, input parsing and validation, and applying rules for prioritization, breaking-up, and grouping of queries.

The Servlet instantiates a JavaBean component that implements the action (model) and invokes its service method (figure 2b). For each service on Services tier contacted, a separate JavaBean component is created and the relevant XML request (figure 3) assigned to it.

The request is then forwarded to a JavaServer Page (JSP) (figure 2b) that represents the View. The JSP extracts the XML response from the JavaBean component, translates it using XTAGS (custom tag library from Apache Software Foundation) [13], and generate an appropriate response to the client (figure 4). XTAGS is implementation of Extensible Stylesheet Language Transformation (XSLT) that is capable of generating HTML, XHTML, and WML output from XML source.

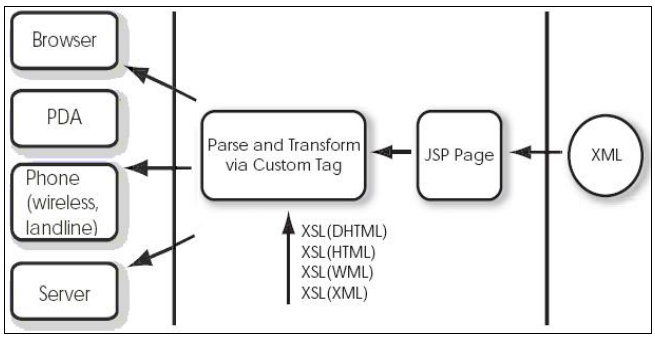

Figure 4: XSLT Transformation.

5.3. Services tier.

All the functionality of the model is concentrated on this tier. It includes different

servers (figure 1), each having specific functionality depending on particular service it offers. A Common feature between all these servers is the way they handle XML queries and format XML responses.

Services working on this tier can be: data logging, plant control and monitoring, commerce services, customers support services, training services, accounting services. The presented paper concentrates on the plant control and monitoring services.

Implementation of these services consists of server (on the services tier) and controllers' network (on the data tier). The server receives XML queries and form transactions to the Data tier. This server is called Transaction server (figure 2c).

For parsing of the XML requests the SAX parser is used. The reason is its event-driven nature, suitable for the model. The Transaction server parses the XML request, extracts queries, and transforms them into transactions of commands. These commands depend on the data access protocol. The results of executed commands are used for the formation of XML responses, based on templates (figure 5).

```
<?xml version="1.0" encoding="UTF-8"?>
<cndep:Response xmlns:cndep=
 "http://net-lab.tu-plovdiv.bg/das"
  xsi:schemaLocation=
  "http://net-lab.tu-plovdiv.bg/das DAS.xsd">
  <controller address="192.168.2.8"
    command="Get Temperature"
    description="Location: lab2104"
    name="IPC@CHIP" port="7777" status="Data">
    <returnValue>
      <Data>
        <context>float</context>
        <value>26.54</value>
      </Data>
    </returnValue>
  </controller>
  <dateAndTime>
    <time>16:23:13</time>
    <date>2006-02-21</date>
  </dateAndTime>
</cndep:Response>
```

Figure 5: XML response.

The actual communication with controllers is based on a custom protocol - CNDEP [11]. This protocol is optimized for monitoring and control of automation devices. It is implemented over UDP/IP. Transaction server sends commands to controllers and receives results. This communication is accomplished through sockets. Commands can be of Get or Set type. The response of a Set command is an acknowledgement or an error message. On a Get command controllers respond with actual data or an error message.

5.4. Data tier.

The Data tier has different expressions for different services on the Services tier. In case of Information services it is a database and in case of Transaction server it is controllers' network (figure 2d). The controllers' network acts as a data producing component. It can implement communication with the controllers in different ways, based on industrial standards or on custom protocols [5].

Each controller in the network can implement monitoring, diagnostic and control tasks. These tasks determine three types of communication – request/response, subscription and spontaneous [8].

The Data tier interacts with the environment by means of sensors and actuators, which implements the actual monitoring and control. The role of the controller is to drive the sensors and actuators and to implement the protocol for data extraction. In current implementation this protocol is CNDEP [11].

6. Conclusions

The presented model is applicable for enterprise systems, allowing integration of business information technologies and automation, or training. It is based on popular multi-tier approach which provides inherited separation of presentation and application logic, security, and reliability. The component approach used, allows interoperability and code reuse.

On the Services tier different functionality can be combined and presented by a single Web portal. The services on this tier can be physically distributed on large distances, which allow centralized control of different plants or factories. Failure of a single server on this tier will affect only the corresponding service, keeping other services available.

Client and Presentation tiers of the model are based on component-of-the-shelf solutions. On the Client tier every web browser can be used. The Presentation tier implementation is based on Java enterprise technology. It can use every Web server supporting Servlets/JSPs (e.g. Sun's Application Server, BEA WebLogic, Apache Tomcat, IBM WebSphere, etc.) and standard web application development tools.

The communication between Presentation and Services tiers is based on XML encoded

messages. This provides the advantages of abstracting away cross-platform communication complexities, increased interoperability between applications, portability for messaging, flexibility, and self-describing data [10].

7. Future work

The future work can go in different directions. One is the integration of web service architecture with the model. The services on the Service tier can be easily implemented as Web services. This will make them dynamically discoverable and suitable for interoperation over large distances. Further, Web services could be provided by controllers from the Data tier.

The other is deeper analysis of the model. This includes long-term analysis of the network traffic and servers' load; using of proxies and service replication; evaluating of different Web servers for the model; etc.

8. Acknowledgements

The work in this paper 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".

9. References

[1] Borriello, G., R. Want, "*Embedded Computation Meets the World Wide Web*", Communications of ACM, Vol. 43 №5, pp. 59-66, May 2000.

[2] Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, *"Pattern-Oriented Software Architecture"*, John Wiley and Sons, 1996, ISBN 0-471-95869-7.

[3] Fortino, G., D. Grimaldi, L. Nigro, *"Distributed measurement patterns based on Java and web tools"*, IEEE Autotestcon Proceedings, pp. 624-628, 22-25, Sep. 1997.

[4] Jazdi, N., *"Component-based and Distributed Web Application for Embedded Systems"*, International Conference on Intelligent Agents, Web Technology and Internet Commerce, 2001.

[5] 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.

[6] Pianegiani, F., D. Macii, P. Carbone, "*An Open Distributed Control and Measurement System Based on Abstract Client-Server Architecture*", IEEE Transactions on Instrumentation and Measurement, Vol 52, Iss 3, pp 686-692, ISSN:0018-9456, Jun 2003

[7] Spasov G., N. Kakanakov, N. Lupanov, *"Three-tier distributed applications"*, Computer Science'2004, 6-8 Dec 2004.

[8] Topp, U., P. Müller, *"Web based service for embedded devices"*, 2001.

[9] Winiecki, W., M. Karkowski, *"A new Java-based software environment for distributed measuring systems design"*, IEEE Transactions on Instrumentation and Measurement, Vol 51, Issue: 6, pp 1340-1346, ISSN: 0018-9456, Dec 2002.

[10] Youngblood, G. M., "Smart Environments", ISBN: 0471544485, Ch 5: "Middleware", pp. 101-127, 2004.

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

[12] http://java.sun.com/products/jsp/ JavaServer Pages.

[13] http://jakarta.apache.org/taglibs/doc/xtags-doc/ XTAG custom tag library.