Journal of Advances in Science and Technology

Vol. IV, No. VII, November-2012, ISSN 2230-9659

REVIEW ARTICLE

FRAMEWORK FOR WEB BASED OPERATING SYSTEMS

www.ignited.in

Journal of Advances in Science and Technology
Vol. IV, No. VII, November-2012, ISSN 2230-9659

Framework for Web Based Operating Systems

Ruchi Agarwal

Research Scholar, Pacific University, Udaipur, Rajasthan, India

----------♦----------

1.1 COMPONENTS OF WBOS

1.1.1 CLIENT/SERVER ARCHITECTURES

In general, web applications have either two-tier or three-tier client/server architectures. The two-tier architecture was developed in the 1980s from the file server software architecture design. Its intention is to improve usability by supporting a form-based user interface. It also improves flexibility and scalability by allocating the two tiers over the computer network. The three-tier (multi-tier) architecture emerged in the 1990s, with a middle tier in-between the user interface and the data management server. This middle tier provides process management and is the place where the business logic and rules are executed. Compared with the two-tier architecture, the multi-tier architecture increases the scalability and flexibility of web applications.

Generally, computers on a network can be categorized into two types: clients and servers. Typically, a client is an application that runs on a personal computer or workstation and relies on a server to perform some operations. A server is a computer or device on a network that manages network resources and provides services. For example, a file server is a computer dedicated to storing files. Any user on the network can store files onto the server. A print server is a computer that manages one or more printers, and a network server is a computer that manages network traffic. This means, machines that provide services to other machines are servers. The machines that connect to those services are clients. For instance, a database server accepts requests for data from clients and returns the results to the clients. The clients manipulate the data and present results to the user.

1.1.2 TWO-TIER ARCHITECTURE

In a two-tier architecture of software systems (see Figure 4.1) server software runs on a large server machine. Client machines connect to the server via a network and make requests of the server as necessary. In this approach, each client machine needs client software installed locally. It works well in relatively homogeneous environments, where application logics (business rules) do not change very often.

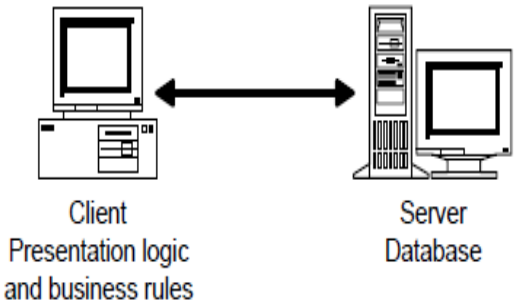

Figure 1.1 : **Two-tier architecture of** Client – Server model

The two-tier architecture improves flexibility and scalability by distributing the two tiers over the network. However, there are obvious limitations with the two-tier software architecture. For example, the client requires a custom application to be written that then needs to be deployed on every client machine. Since the presentation logic and business rules are usually located in the client application, even the smallest change to an application might require a complete rollout to the entire system.

1.1.3 THREE-TIER ARCHITECTURE

The three-tier architecture consists of three well-defined and separate processes, each running on a different platform as shown in Figure 4.2 . The first tier is referred to as the user interface, which runs on the user's computer (the client). The middle tier consists of the application or business logic, which runs on a server and is often called application server. The other tier contains the data that is needed for the application, which is usually a database management system (DBMS) that stores the data required by the middle tier.

This tier runs on another server called the database server. The three-tier design has many advantages over the traditional two-tier system. For example, separating presentation logic from business rules

makes it easier to modify or replace one tier without affecting the others.

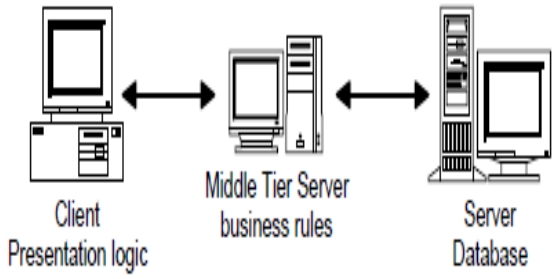

Figure 1.2 : **Three tier architecture of** Client – Server model

The three-tier architecture has been used successfully since the early 1990s in commercial and military distributed client/server environments, where distributed information computing is required in a heterogeneous environment.

1.2 ABSTRACT MODEL OF WBOS

WBOS is a multi-tier web application using CORBA as a distributed object framework. This means that it is designed and implemented using a multi-tier client/server architecture with distributed object technology. Unlike many other web applications, WBOS is a specific web application for industrial process control and production monitoring. Therefore, the architecture of WBOS is different from those of generic web applications even though it adopts some of the existing technologies, such as multi-tier client/server architectures, and distributed object technologies.

1.2.1 OVERVIEW OF STRUCTURE

An overview of the application architecture is shown in Figure 4.3. WBOS is viewed as a collection of computer nodes that are communicating and cooperating to reach a common goal. The nodes in WBOS are geographically dispersed across the Internet/intranet.

Nodes can be homogeneous or heterogeneous. We adopted the heterogeneous architecture, because nodes in WBDCS may have different hardware architectures and software configurations, such as PCs running Windows and Sun workstation running Solaris.

The devices that are connected to nodes may be different such as controllers, data acquisition systems, radio remote control devices, OS, and DBMS.

A homogeneous system can be considered a special case of the heterogeneous systems.

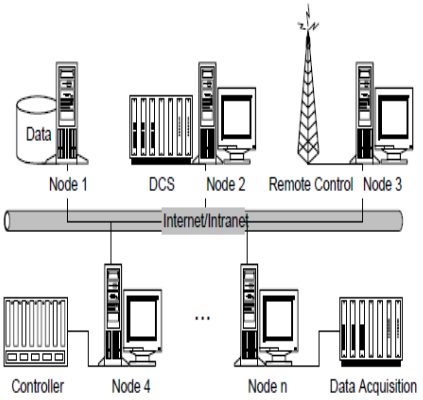

Figure 1.3 : Application architecture of WBOS

The system also allows users to integrate their existing applications, such as PLC, DCS, into a web-enabled distributed system by wrapping them with CORBA objects. These objects can then make calls to legacy systems and expose them to the Internet/intranet.

NODES

The nodes in WBDCS are PCs and/or workstations connected to one or more devices. Each node may have different hardware and software configurations with different devices connected. This allows WBDCS to be a flexible heterogeneous system. All nodes in WBDCS can communicate with each other across the Internet/intranet. A node can be a client when it is sending a request to another node; on the other hand, it can also be a server when it provides a service to other nodes in the system. A client node does not need to install any client application in order to be able to access any other nodes because the system is designed using applets as GUIs that are automatically downloaded from a web server in WBDCS.

1.2.2 THE ARCHITECTURE OF WBOS

WBDCS is designed using a web-based multi-tier client/server software architecture and CORBA technology. As mentioned in earlier section , two-tier architectures have problems with maintainability. The need to install the client application on every client machine can be costly depending on how many clients there are and how often updates will be made. The web-based multi-tier distributed object architecture attempts to address this issue.

1.2.3 WEB-BASED MULTI-TIER APPROACH

WBOS is designed using a multi-tier client/server software architecture with a web application approach. It consists of a web server and an IIOP gateway, control servers, devices, and a database. Its architecture is depicted in Figures 1.5 and 1.4 using different points of view. It is a multi-tier web

application, which includes a client tier, a middle tier, and a data tier as illustrated in Figure 1.2 .

The web server (Apache HTTP Server or VisiBroker Gatekeeper) is where the applets are located. The IIOP gateway (VisiBroker Gatekeeper) is an OMG-CORBA compliant GIOP proxy server, which enables CORBA clients and servers to communicate across networks, while still conforming to security restrictions imposed by Internet browsers, firewalls and Java sandbox security.

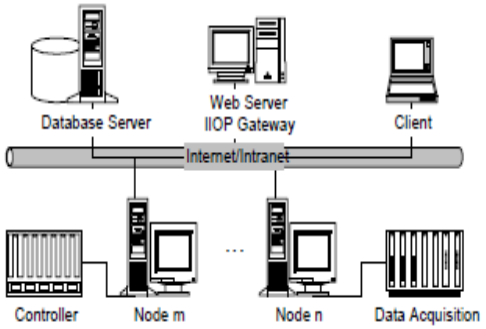

Figure 1.4 : WBOS **Architecture**

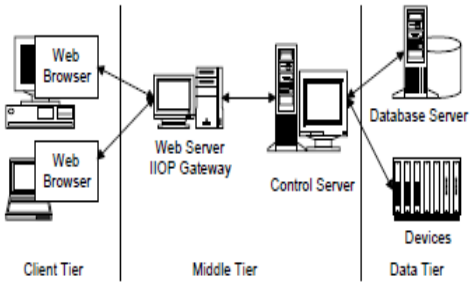

Figure 1.5 : 3 tier Architecture of WBOS

1.3 3 TIER ARCHITECTURE OF WBOS

The control servers are usually located in nodes. They are the middle tier between client and data tiers. They provide two parts of services. One is control service; the other is information management service. The control service includes sampling, processing and responding services. The information management service includes storing real-time data into the database and updating real-time data on client GUIs in a certain period of time.

A user may access the application by navigating to the node's URL using a web browser on a client machine. Applets are downloaded from the web server and run in a user's browser. The prime advantage of this web-based approach is that all code associated with the client tier is downloaded dynamically from the web server. There is usually no need to install any application-specific software on the client machine. This greatly reduces long-term maintenance costs.

1.3.1 DISTRIBUTED OBJECT APPROACH

A distributed object system is a system in which all entities are modeled as objects. It is a popular paradigm for object-oriented distributed applications. CORBA is a standard framework for distributed object systems. It allows a distributed, heterogeneous collection of objects to interoperate across a network.

Distributed object technology allows large server programs to be broken down into several smaller server objects. Each object can potentially reside on a different machine on the network. Server objects can even be run on small desktop computers rather than on a large server machine. Since distributed objects allow applications to be split into lightweight pieces that can be executed on separate machines, less powerful machines can run heavily demanding applications [8]

1.3.2 ANALYSIS OF THE APPROACHES

The growing popularity of distributed object technologies has been driven by several problems with the two-tier client/server approach, which include the following:

Scalability - This term refers to how easily a particular solution can be extended from a small-scale application to a large-scale one. Many applications work well with just a few functions and users, but fall apart when having to support large numbers of functions and users. Also, some applications perform well on a Local Area Network (LAN), but may not work well on a Wide Area Network (WAN).

Maintainability - This term refers to how costly it is to administer a particular solution, which includes costs associated with updating the software and distributing updated versions to client machines. By using web-based distributed object technologies, the above problems can be minimized and sometimes eliminated. There is usually no need to design complex infrastructure software that improves scalability and maintainability: the necessary infrastructure already exists, and web-based distributed object technologies take advantage of that fact.

As described above, the advantages of using a web-based distributed object solution are obvious. However, there are also disadvantages compared to a two-tier client/server solution. The two-tier solution is the simplest approach: client applications talk directly to servers. The web-based distributed object approach requires the use of web servers, web browsers, and intermediate server objects. The extra

layers of software improve the scalability and maintainability of the system, but at the cost of simplicity. There are also disadvantages compared to a non web-based distributed object solution. The primary means of dynamically downloading code into a browser used in this system are Java applets. Using Java applets introduces the following disadvantages:

Security - Running an applet using a browser has certain security restrictions. Therefore, most dynamically downloaded applet code cannot do all of the things that a standalone client application can do. This issue will be discussed in the following subsection.

Performance during initialization - Because client application code must be downloaded from a web server, initialization time is much longer than that of a standalone application.

Increased network traffic - Downloading an applet (thin client) from a web server usually increases network traffic compared to a standalone version of the same client application installed on a client machine. But it might not increase the traffic if a client application contains both presentation logic and business rules like the two-tier client/server solution in section 4.1.

Run-time performance - Java “bytecode” was designed to work in any browser on any platform. It must be interpreted by the JVM in a browser, and converted to native machine instructions. This conversion process could potentially decrease performance of applications. Standalone applications are usually in the form of compiled code, which requires no interpreter. In most cases, these disadvantages are acceptable given the great number of advantages.

1.3.3 APPLET SECURITY ISSUES

Applets are used as client GUIs by dynamically downloading from the web server in WBOS. In addition to the disadvantages mentioned above, there are some primary security restrictions imposed on Java applets such as file I/O, printing and network access. It is referred to as the Java “Sandboxing” security model, which limits the applets’ effectiveness in distributed object applications. Network access restrictions are as follows:

- An applet can only establish network connections with the host that served the applet.
- An applet can only accept network connections from the host that served the applet.

In other words, Java “Sandboxing” security prevents Java applets from communicating with server objects located on servers other than the ones running on the host from which the applets were downloaded.

These network restrictions create a big problem for CORBA application. CORBA provides location transparency, that is, as long as a client holds an Interoperable Object Reference (IOR), it can invoke operations on a server object, regardless of the location of the server object. Applet sandboxing breaks CORBA location transparency [1].

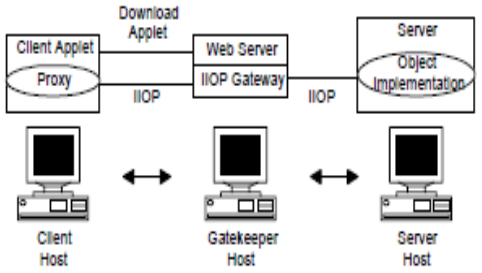

Figure 1.6 : IIOP gateway model for WBOS

These restrictions can be solved by using an IIOP gateway on a web server as illustrated in Figure 4.6, such as VisiBroker Gatekeeper. The IIOP gateway acts as a proxy for the CORBA object by sending requests to the object and passing responses back to the applets. Without the IIOP gateway, Java applets will only be able to use references to objects that reside on the web server host.

1.4 THE DESIGN OF WBOS

The software architecture of WBOS is not only concerned with structure and behaviour, but also with usability and functionality. By using UML, the system architecture can be modeled from different perspectives in order to visualize, specify, construct, and document the system.

1.4.1 FUNCTIONAL DESCRIPTIONS

The use cases of the system describe aspects of behaviour of the system as seen by its end users or testers. UML allows the static aspects of the system to be captured in use case diagrams and the dynamic aspects of the system to be captured in activity diagrams.

A use case diagram [9] is a description of a set of sequences of actions. An actor represents a coherent set of roles that users of use cases play when interacting with these use cases. Typically, an actor represents a role that a human, a hardware device, or even another system plays.

An activity diagram [9] is one kind of UML diagram used for modeling the dynamic aspects of a system. It emphasizes the flow of control from activity to activity.

An activity is an ongoing non-atomic execution within a state machine. Activities ultimately result in some

action, which is made up of excitable atomic computations that result in a change in state of the system or the return of a value.

As illustrated in Figure 1.7, there are four actors in WBDCS. The user represents a role that interacts with the system. The database is an information management system, which stores the configuration and operation information of the system. The sensor is a device that measures process variables from a physical environment. The actuator is a device that performs an action towards the physical environment.

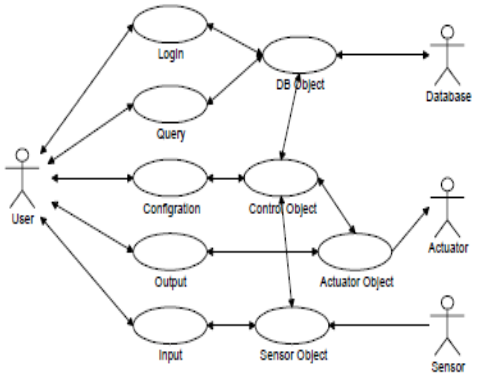

Figure 1.7 : Use case diagram for WBOS

Sensor, control, actuator, and DB objects are made up of CORBA server objects called the control server tier in WBDCS. The sensor object is responsible for acquiring data from the sensor, passing them to the control object and user. The actuator object is responsible for writing data from the user or control object to the actuator. The control object is a controller that processes sampled data and produces a response to the actuator object. The DB object is responsible for collecting data from the control object and for storing them in a database.

Login, query, configuration, output, and input are the client tier in WBDCS. Five use cases have been identified in Figure 4.7. These use cases will be described in detail in the following subsections.

1.4.1.1 LOGIN

In this use case, a user has to login the system before using it. The user's information will be verified to make sure that the user has the authority to use the system.

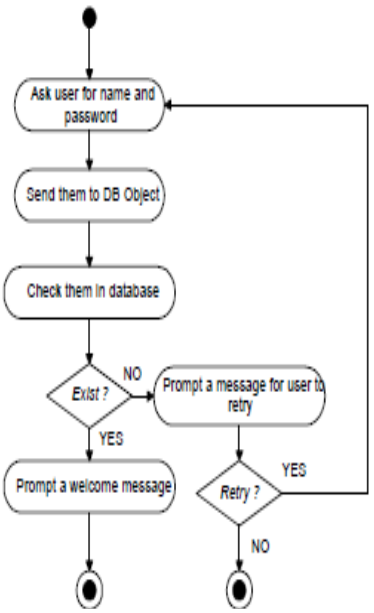

Figure 1.8 : Login Activity flow in WBOS

In Figures 1.8, a user is the actor who initializes the use case. The user name and password will be sent to the DB server object via the network, and the DB object will check if they are the same as the ones stored in the database. The DB object will notify the user about the result of the check in one of two possible ways: one is the welcome information, which means the user information is valid; the other is a warning message, which means that user name and/or password were wrong.

1.4.1.2 INPUT

The input process, as depicted in Figures 1.9 and 1.10, is defined as reading data from the sensor. In this use case, the user will first select the input channels and then send an input request to the sensor object. The sensor object will get the data from sensor and send them back to the user.

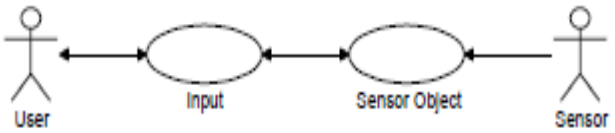

Figure 1.9: The input process

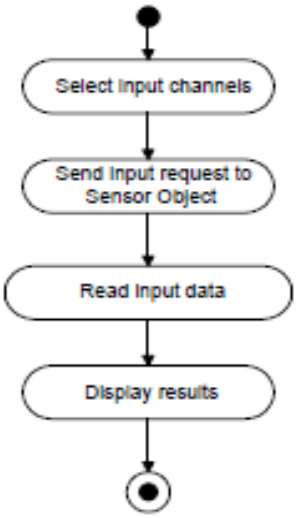

Figure 1.10: Input activity diagram

1.4.1.3 OUTPUT

The output process, as illustrated in Figures 4.11 and 4.12, is defined as writing data to an actuator. In this use case, the user first selects the output channels and sets the output values, and then sends an output request to the actuator object, which will write the output data to the actuator.

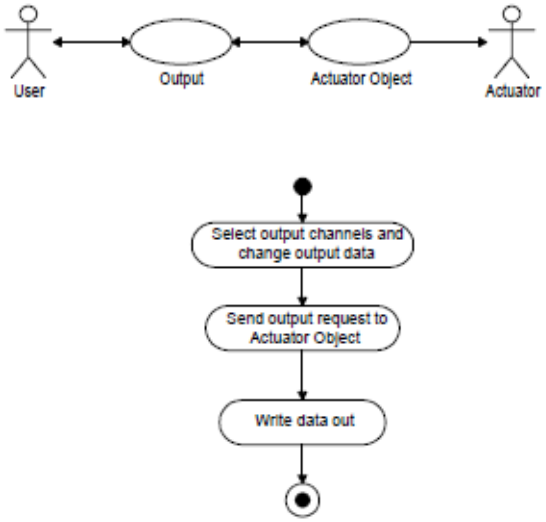

Figure 1.11 and 1.12 : Output process diagrams

1.4.1.4 QUERY

The querying process, as shown in Figures 1.13 and 1.14, is defined as querying historical data from the database. In this use case, the user will first select information to be searched, and then the request is sent to the DB server object. The DB object will access the database, find the requested data, and send them back to the user.

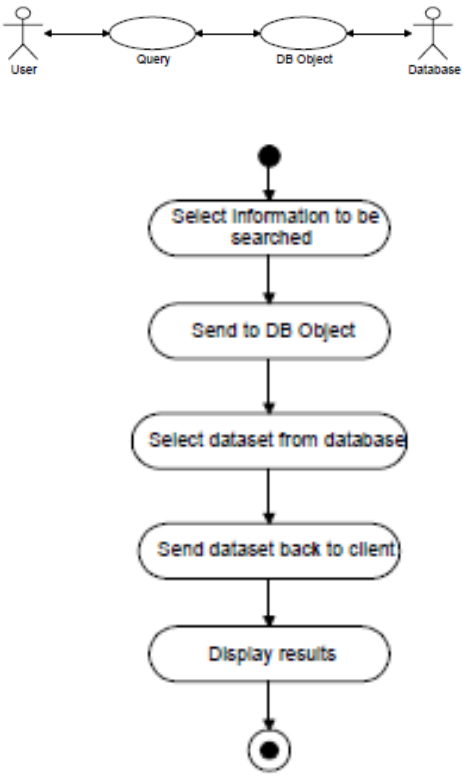

Figure 1.13 and 1.14 : Query process diagrams

1.4.1.5 CONFIGURATION

The configuration process is defined as configuring the working modes of the controller (control object) and making the control loop work properly as depicted in Figures 4.15 and 4.16. In this use case, the user can select control algorithms, change set point (a desired value for a controlled variable), and switch the controller from manual to automatic mode, and vice versa.

In automatic mode, the controller will cooperate with the sensor, actuator, and DB objects to ensure that a physical process is controlled successfully. The sensor object samples the process variable (*PV*) measured by the sensor and passes it to the control object. The control object compares the *PV* with a desired value or set point (*SP*) and produces an output using a certain control algorithm, and sends the output to the actuator object. The actuator object manipulates the actuator according to the output of the controller. These control procedures execute continuously to maintain the physical process under control. The DB object will store the configuration and operation information in the database.

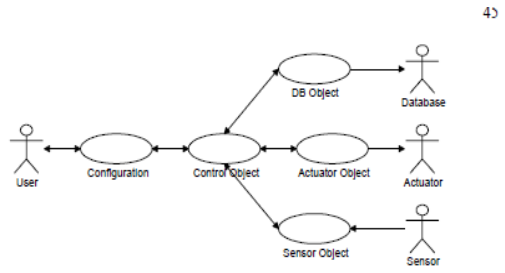

Figure 1.15 : Configuration process use case diagrams

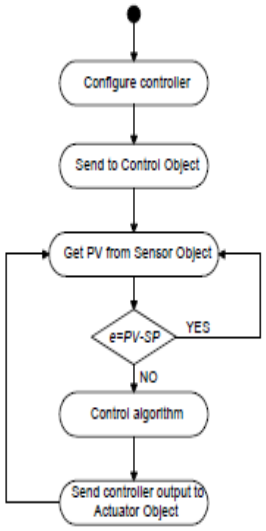

Figure 1.16 : Configuration process flow diagrams

1.4.2 CONTROL ALGORITHMS

There are many existing control algorithms that can be adopted and implemented in the controller (control object) of WBDCS. In the following subsections, we will introduce two of the most popular control algorithms.

1.4.2.1 PID CONTROL

PID (Proportional + Integral + Derivative) is a well-established control algorithm, which is commonly used in process industry. A single-input and single output feedback control system consists of a sensor, a controller, an actuator and a process as illustrated in Figure 1.19. The goal of this loop is to maintain the level in the tank at certain value. *SP* is a predetermined value for the level. *PV* is the actual level measured by the sensor. Again, the control loop is to maintain *PV* at a predetermined *SP*.

ON-OFF CONTROL

The most rudimentary form of regulatory control is on-off control [7]. An example of on-off control is a home heating system. Wherever the temperature goes above the set point, the heating system shuts off, and the temperature drops below the set point, the heat system turns on. This control algorithm is shown by Equation).

$m(t) = 0\ \%$ if $PV > SP$

$m(t) = 100\ \%$ if $PV < SP$

The controller output is equal to 0% whenever *PV* exceeds *SP*. The controller output is 100% whenever *PV* is below *SP*. The ideal dynamic response of the on-off control loop is depicted. For simplicity, hysteresis is not considered in the example.

There are different control algorithms for different physical environments. WBOS allows the implementation of different control algorithms in the control object, i.e., the system allows different controller to be implemented.

VIRTUAL DEVICES

Virtual devices in WBOS refer to any simulated process devices depicted on a web page. Those devices are made up of an imitated process on a HTML page, and are the miniature of actual devices. Virtual devices are clickable when the user needs to know more about them and to operate them.

Figure 1.17 illustrates the idea of virtual devices as proposed in WBDCS. It is one of the GUIs (clients) for a level control loop. All devices on the page may be linked to other pages to retrieve detailed information related to these devices. The table on the page is an applet that displays the real-time operational data of the actual control loop. If a real process is under control, the *PV* will always track the change of *SP* and keep steady state offset within an acceptable limit. Otherwise, the process will be out of control. In this case, *PV* may increase until it reaches its highest limit, a high alarm will be sounded, and a status icon in the table will turn red. In such a case, an operator may click the controller and switch the controller from automatic mode to manual mode, open the valve to 100%, and force the level to go down.

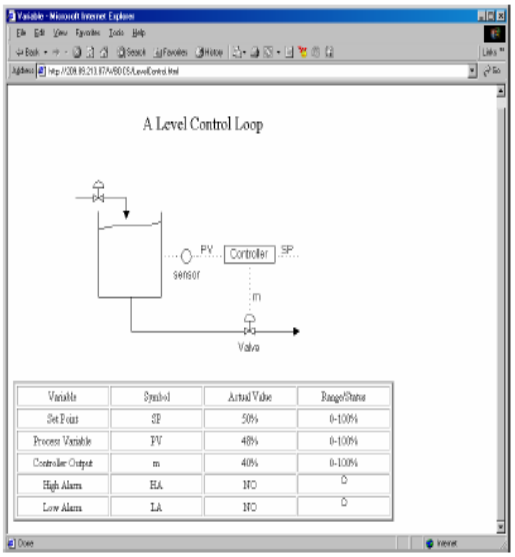

Figure 1.17: Virtualization process in WBOS

REFERENCES

[1] M. Janke, “OPC-simple software integration for legacy systems”, *IEEE Industry Applications Society Advanced Process Control Applications for Industry Workshop*, 1999.

[2] Weonjoon Kang; Hyoungyuk Kim; Hong Seong Park, “Design and performance analysis of middleware-based distributed control systems”, *Proceedings of 2001 8th IEEE International Conference on Emerging Technologies and Factory Automation*, Volume: 2, 2001.

[3] D.I. Katcher, H. Arakawa and J.K. Strosnider, “Engineering and Analysis of Fixed Priority Schedulers,” *IEEE Transactions on Software Engineering*, 19(9), September, 1993.

[4] J. Lehoczky, L. Sha, and Y. Ding, “The rate monotonic scheduling algorithm: exact characterization and average case behaviour,” *Proceedings of 10th IEEE Real-Time Systems Symposium*, Santa Monica, CA, December 1989.

[5] Scott M. Lewandowski, “Frameworks for Component-Based Client/Server Computing”, *ACM Computing Surveys*, Vol.30, No.1, March 1998.

[6] R. Lewis, “Design of distributed control systems in the next millennium”, *Computing & Control Engineering Journal*, Volume: 8 Issue: 4, August 1997.

[7] C.L. Liu, and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard real time environment,” *Journal of the Association for Computing Machinery*, v.20, n.1, pp. 44-61, January 1973.

[8] Yih Ping Luh, Shean-Shyong Chiou, Jau-Woie Chang, “Design of distributed control system software using client-server architecture”, *Proceedings of The IEEE International Conference on Industrial Technology*, 1996.

[9] P. Marti, J.M. Fuertes, G. Fohler, “An integrated approach to real-time distributed control systems over fieldbuses”, *Proceedings of 2001 8th IEEE International Conference on Emerging Technologies and Factory Automation*, Volume: 1, 2001.

[10] J.M. Nogiec, E. Desavouret, D. Orris, J. Pachnik, S. Sharonov, J.C. Tompkins, K. Trombly-Freytag, “A distributed monitoring and control system”, *Proceedings of the 1997 Particle Accelerator Conference*, Volume: 3, 1998.

[11] Object Management Group, Specification of the Portable Object Adapter (POA), OMG Document orbos/97-05-15 ed., June 1997.

[12] Object Management Group, The Common Object Request Broker: Architecture and Specification, 2.2 ed., Feb. 1998.

[13] Object Management Group, Realtime CORBA, OMG TC Document orbos/98-10-05.

[14] http://www.opcfoundation.org/default.asp[Selic99] B. Selic, “Turning Clockwise: Using UML in the Real-Time Domain”, *communication of the ACM*, Vol.42, No.10, October 1999.

[15] L. Sha and J. B. Goodenough, "Real-Time Scheduling Theory and Ada," *IEEE Computer*, April 1990.

[16] L.H. Stolen, “Distributed Control System”, *INTELEC '99., The 21st International Telecommunication Energy Conference*, 1999.

[17] William Y. Svrcek, Donald P. Mahoney, Brent R. Young, “A Real-Time Approach to Process Control”, *John Wiley & Sons Ltd.*, ISBN: 0-471-80363-5, 2000.

[18] Kok Kiong Tan, Tong Heng Lee, Chai Yee Soh, “Remotely operated experiment for mechatronics: monitoring of DCS on the Internet”, *Proceedings. 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Volume: 2, 2001.

[19] Kok Kiong Tan, Tong Heng Lee, Chai Yee Soh, “Internet-based monitoring of distributed control systems-An undergraduate experiment”, *IEEE Transactions on Education*, Volume: 45 Issue: 2, May 2002.

[20] Stephen A. Thomas, "SSL and TLS Essentials", *John Wiley & Sons Inc*., ISBN: 0-471-38354-6, 2000.

[21] Jeffrey J.P. Tsai, Yaodong Bi, Steve J.H. Yang, Ross A.W. Smith, "Distributed real-time systems: monitoring, visualization, debugging, and analysis", *John Wiley & Sons Inc.,* ISBN: 0471160075, 1996.

[22] Steve Vinoski "CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments", *IEEE Communications* 14, 2, February 1997.

[23] Yirong Yang, Shanan Zhu, "Small smart distributed control system", *Proceedings of the 4th World Congress on Intelligent Control and Automation,* Volume: 3, 2002.