Design of a Simple Service Oriented Supervisory Control and Data Acquisition System

Nedim Osmić, Jasmin Velagić
Faculty of Electrical Engineering/University of Sarajevo
Sarajevo, Bosnia and Herzegovina
nedim.osmic@etf.unsa.ba, *jasmin.velagic@etf.unsa.ba*

***Abstract*—The goal of this paper was to design a simple Service Oriented Supervisory Control and Data Acquisition System (SCADA) that can be used to manage, control and visualize multiple heterogeneous SCADA systems. A three-tier System architecture was designed consisting of Data Access, Service and Presentation Layers. Clients on the presentation layer can communicate with the system backend via SOAP XML or RESTful JSON Web services. The full entity relationship diagram of the database schema is presented in the paper. Message sequence charts are explained in detail for polling and event based notification of setpoint changes. At the end of the paper a simple Web user interface was introduced to demonstrate how modern Web technologies like AJAX and JQuery can be used to build interactive SCADA user interfaces.**

***Keywords*—SCADA system; Data acquisition; Interactive user interface; SOAP XML; Control system topology**

I. Introduction

With the emerge of Web technologies, Service oriented architecture has become a dominant software architecture model in industrial software solutions. In order to ensure interoperability between devices and Web services vendors manufacture modern instruments that support Web formats such as XML and other related open standards [1]. A possible migration strategy from current solutions to a Service oriented architecture is presented in [2].

Traditional SCADA systems were used for intracompany information exchange due to low bandwith and communication constraints that were present at that time. Technological advances in networking have made it possible to fast, secure and cost effective share informations of multiple systems over the internet. Even real time internet communication is not a limiting factor anymore for most industrial applications [3]. Web based solutions have been accepted by academic and industrial communities alike [3], [4]. The evolution of SCADA systems from traditional to Next generation systems is described in [5].

Lately, another emerging research topic concerning Service oriented SCADA systems are possible expansions to Cloud based solutions that can provide a reliable service backend for large scale SCADA systems. Important factors and possible problems that have to be considered in this expansion are presented in papers [6] and [7].

This paper proposes a design to create a Service oriented SCADA software system that provides mandatory SCADA functionality via SOAP XML and RESTful JSON Web services as interface. Based on the experience from papers [8] and [9] Java was chosen as platform due its reliability as server side technology. Other advantages of Java is that its free, well documented and has a strong user community.

The paper is organized as follows. In section 2 the system architecture is described. Detailed information about the database schema is given in section 3. Implemented Web services and their description is given in section 4. In section 5 we introduced a ASP.NET MVC3 based Web interface to demonstrate how rich user interfaces can be built on top of the designed system. Conclusions and further research guidelines are discussed in the final section.

II. System architecture

The systems architecture is shown in Fig. 1. It represents a typical three tier client server-architecture consisting of common Data Access (persistence) and Service (application logic) layers while independent groups of clients implement Presentation layers on their own. This gives system designers the opportunity to tailor their SCADA interfaces to their needs while sharing mutual database and application server resources.

The Data Access Layer consists of Plain Old Java Objects (POJO's) that represent the data from the database tables and the Data Access Object (DAO) which serves as an Application Programming Interface (API) between POJO's and database tables. The DAO was written on top of the widely accepted Java Database Connectivity (JDBC) API and provides the necessary database interface functionalities.

The Service Layer contains the DAO Service Wrapper which serves as a data wrapper between POJO's from the Database Access Layer and the Service Layer. It also delegates Web Service method invokations to their respective functions in the DAO Persistence API to achieve the desired functionality. The Service Layer also contains both a SOAP and Java Simple Ocject Notation (JSON) RESTful Service API to give system designers the capability to achieve advantages of each service architecture. For instance, SOAP services could be used to ensure reliable measurement storage for slow industrial processes, while RESTful services could be used for fast measurement retrievement necessary for user interface components that require real time graphical representation of the process dynamics. The key advantages of using SOAP services as SCADA backend are described in [10].

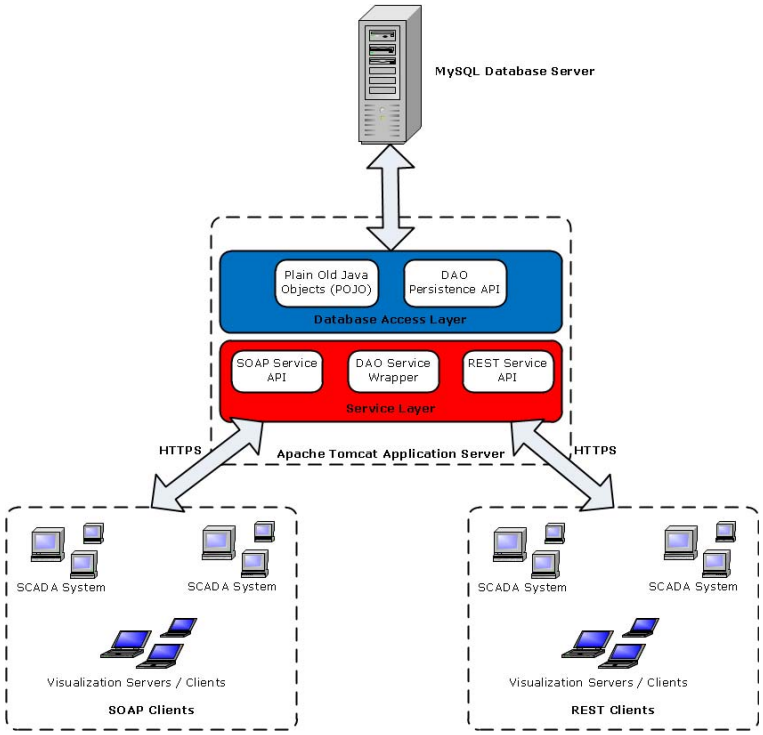

Figure 1: Service Oriented SCADA Architecture

In order to describe the system architecture in details it is crucial to analyze the database structure.

III. Database structure

The Entity Relationship Diagram (ERD) of the database is shown in Fig. 2. The tables can be divided into 4 groups: User, system configuration, system representation and data tables.

The User table defines all system users and their roles. Possible roles are ScadaUser and ScadaAdmin. A ScadaUser can only monitor the data from the system (pull measurements) while a ScadaAdmin can configure the whole system.

Each user can create Scada nodes that are defined in the ScadaNode table. Scada nodes represent master devices from industrial master-slave configurations and each node can have its own slave devices defined in the NodeSlaves table. Slave devices have sensors that are defined and configured in the SlaveSensor table. It is required to set up the sensor characteristic. These tables including the ScadaNodeListener table represent the system configuration tables. The ScadaNodeListener table is a special table that gives SCADA nodes the possibility to receive asynchronous setpoint changes. This functionality is discussed in details in the next section.

System representation tables are used to create a representational hierarchy of the SCADA system. SCADA nodes can be grouped into workspaces, and workspaces can be bound to groups. The Group table creates a tree hierarchy based on its entries. For instance, an example system hierarchy is shown in Fig. 3. It consists of three groups: Plant A, Process A and Process B. Plant A is the parent group of Process A and Process B. Both child groups have one workspace attached to it, each with two nodes: Node A and Node B.

Data tables are solely used for data storage and retrieval. Sensor measurements are stored in the Measurement table and setpoint history is stored in the SetpointHistory table. The data from these tables can be used for real time graphical representation of the process dynamics.

IV. Web services

The list of hosted Web services and their respective methods is shown in Table I. Most Web services are create, read, update and delete actions and do not require further discussion.

The system supports two different ways to acquire setpoint changes based on the restrictions of typical SCADA hardware [11]. The first approach is polling based and its message sequence chart (MSC) is shown in Fig. 4. Polling is acceptable in situations where its not possible to implement a listener Web service on the SCADA node due to hardware constraints.

The other approach uses a similar subscriber / publisher pattern described in [12] that is used for SMS status tracking. The only difference being that the listener is JSON RESTful service and not a SOAP service. The SCADA user subscribes the SCADA node into the ScadaNodeListener table of the database via the subscribeListener method. Unsubscription is achieved via the unsubscribeListener method. MSCs of those actions are shown in Fig. 5 and Fig. 6, respectively. MSC from Fig. 7 illustrates how a user cause node setpoint change is reflected interactively via the node listener.

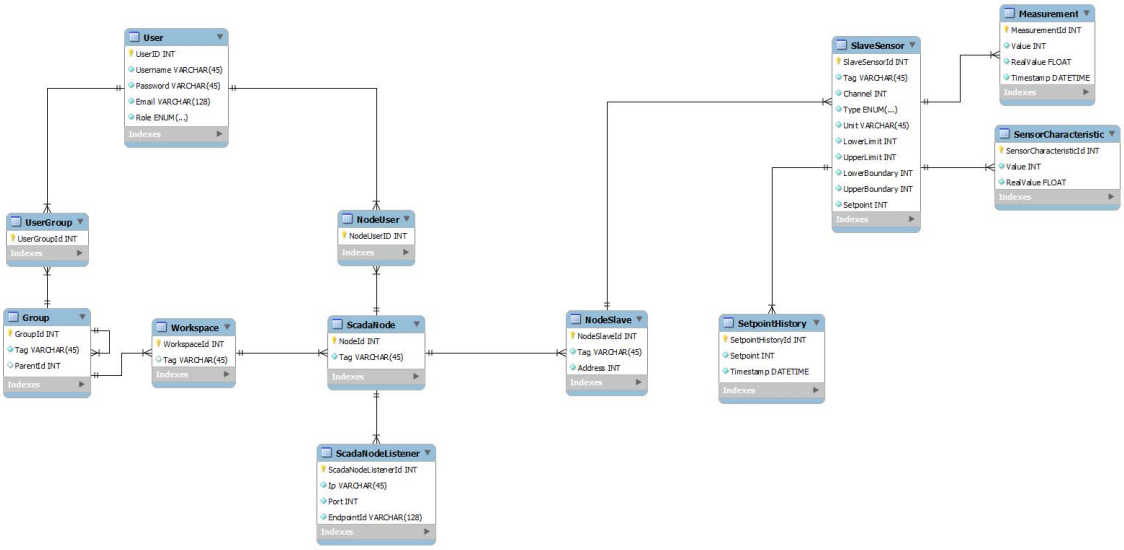

Figure 2: Entity relationship diagram of the database schema

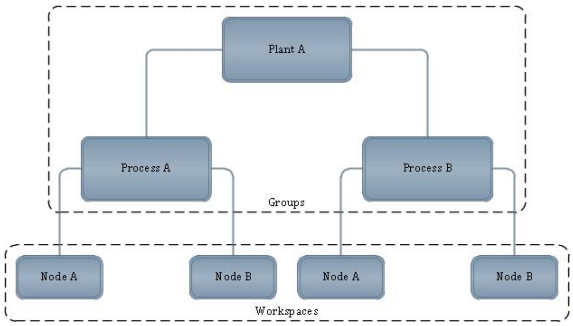

Figure 3: System hierarchy Example

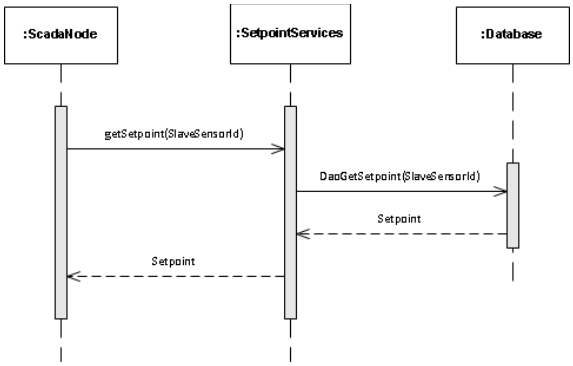

Figure 4: MSC for setpoint polling

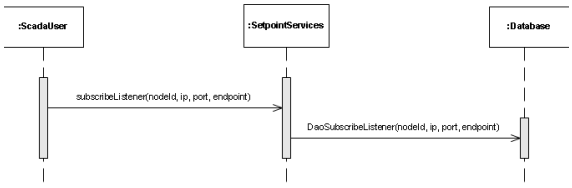

Figure 5: MSC for listener subscription

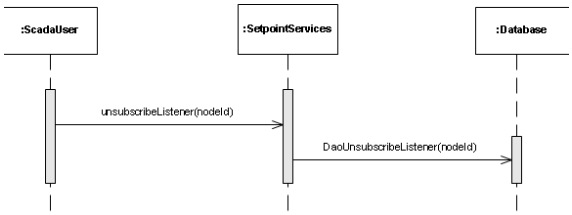

Figure 6: MSC for listener unsubscription

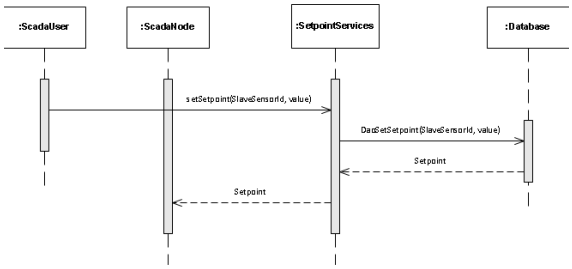

Figure 7: MSC for setpoint notification

V. Example Client Implementation

In order to test the designed system a ASP.NET MVC3 Web application was written for a simple SCADA system consisting of the following components: Modicon PLC, induction motor, Altivar invertor, motor sensors (speed, torque and current), two Twido PLCs and two light bulbs. Visualization of the SCADA systems topology, written using arbor.js, is given in Fig. 8. The topology supports interactive callbacks via Asynchronous JavaScript and XML (AJAX), i.e. bulb states can be changed on click. AJAX is a neccessary technology to build rich desktop like Web based SCADA user interfaces [13]. In order to achieve interactive AJAX callbacks it was necessary to implement a service controller that serves as a

proxy for cross domain AJAX calls.

VI. Conclusions

This paper presented a design approach to implement a simple Service oriented SCADA system. The proposed design is based on a stack multitier software architecture containing both SOAP XML and RESTful JSON Web service interfaces. Detailed database ERD schema and implemented Web services MSC were given. A rich ASP.NET MVC3 based Web interface was built on top of the designed system to demonstrate the functionality of the system.

In further research we will analyze the scalibity and reliability of the developed system by using it to drive all SCADA systems that exist at our department.

TABLE I: SCADA system available Web services

Service name	Service functions
NodeServices	• addNode • removeNode • editNodeTag
NodeListenerServices	• subscribeListener • unsubscribeListener
NodeSlaveServices	• addNodeSlave • removeNodeSlave • editNodeSlaveTag • editNodeSlaveAddress
SlaveSensorServices	• addNodeSlaveSensor • removeNodeSlaveSensor • editNodeSlaveSensorTag • editNodeSlaveSen.Channel • editNodeSlaveSen.Type • editNodeSlaveSen.Unit • editNodeSlaveSen.LowerLimit • editNodeSlaveSen.UpperLimit • editNodeSlaveSen.LowerBound • editNodeSlaveSen.UpperBound • editNodeSlaveSensorSetPoint
SlaveSensorCharacteristicServices	• addCharacteristicPoint • removeCharacteristicPoint • editCharacteristicPoint
WorkspaceServices	• addWorkspace • removeWorkspace • editWorkspaceTag • addScadaNodeToWorkspace • removeScadaNodeFromWorks.
GroupServices	• addParentGroup • removeParentGroup • addChildGroup • removeChildGroup • addWorkspaceToGroup • removeWorkspaceFromGroup • editWorkspaceGroup
MeasurementServices	• addMeasurement • addMeasurementBatch • getMeasurementsFromInterval • getLastMeasurements
SetpointServices	• changeSetpoint • getSetpoint • getSetpointsFromInterval • getLastSetpoints • getAllSetpointsForScadaNode • getAllSetpointsForNodeInterval • getAllLastSetpointsForNode

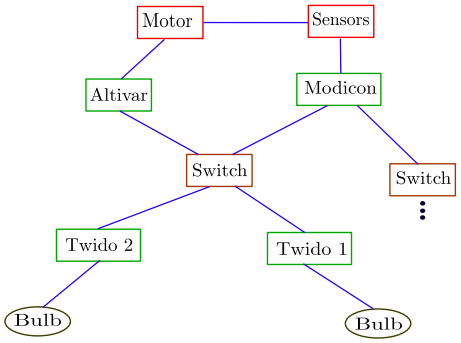

Figure 8: Example topology

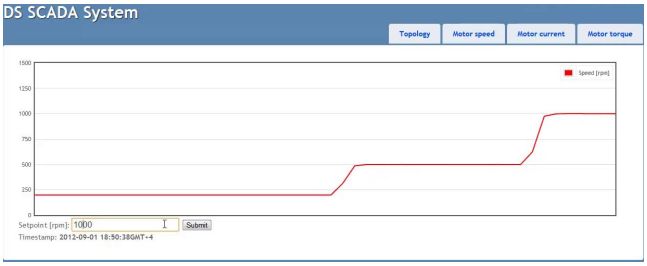

Figure 9: Motor speed visualization

References

[1] R. Rutkauskas, A. Lipnickas, C. Ramonas and V. Kubilius, "New challenges for interoperability of control systems," *Electronics and Electrical Engineering*, vol. 83, no. 3, pp. 71-74, 2008.

[2] J. Delsing, F. Rosenqvist, O. Carlsson, A. W. Colombo and T. Bangemann, "Migration of industrial process control systems into service oriented architecture," In Proc. IEEE IECON Conference, 2012, pp. 5786-5792.

[3] B. Qiu, H. B. Gooi, Y. Liu and E. K. Chan, "Internet-based SCADA display system," *IEEE Computer Applications in Power*, vol. 15, no. 1, pp. 14-19, 2002.

[4] Inductive automation, White Paper, Cloud-Based SCADA Systems: The Benefits & Risks, 2011.

[5] S. Karnouskos and A. W. Colombo, "Architecting the next generation of service-based SCADA/DCS system of systems," In Proc. IEEE IECON Conference, 2011, pp. 359-364.

[6] S. Karnouskos, A. W. Colombo, T. Bangemann, K. Manninen, R. Camp, M. Tilly, P. Stluka, F. Jammes, J. Delsing and J. Eliasson, "A SOA-based architecture for empowering future collaborative cloud-based industrial automation," In Proc. IEEE IECON Conference, 2012, pp. 1-8.

[7] F. Jammes, B. Bony, P. Nappey, A. W. Colombo, J. Delsing, J. Eliasson, R. Kyusakov, S. Karnouskos, P. Stluka and M. Tilly, "Technologies for SOA-based distributed large scale process monitoring and control systems," In Proc. IEEE IECON Conference, 2012.

[8] A. F. Ramadan, L. Cheded and O. Toker, "Improving internet-based SCADA systems using Java and XML," Unpublished paper.

[9] R. Fan, L. Cheded and O. Toker, "Internet-based SCADA: a new approach using Java and XML," *Computing and Control Engineering*, vol. 16, no. 5, pp. 22-26, 2005.

[10] Q. Chen, H. Ghenniwa and W. Shen, "Web-services infrastructure for information integration in power systems," In Proc. Power Engineering Society General Meeting, 2006.

[11] A. Martinic, "SCADA systems in heterogeneous environments," In Proc. MIPRO International Convention, 2005, pp. 1-6.

[12] The European Telecommunications Standards Institute, ETSI OSA Parlay X version 2.1 standard specification Part 4: Short Messaging (ETSI ES 202 391-4 V1.2.1), December 2006.

[13] A. M. Mohamed and H. Abbas, "Efficient web based monitoring and control system," In Proc. International Conference on Autonomic and Autonomous Systems, 2011, pp. 18-23.