Proposing a Novel Architecture of Script Component to Incorporate the Scripting Language Support in SCADA Systems

Muhammad Waseem Anwar, Farooque Azam

▶ **To cite this version:**

Muhammad Waseem Anwar, Farooque Azam. Proposing a Novel Architecture of Script Component to Incorporate the Scripting Language Support in SCADA Systems. 13th IFIP International Conference on Computer Information Systems and Industrial Management (CISIM), Nov 2014, Ho Chi Minh City, Vietnam. pp.351-362, 10.1007/978-3-662-45237-0_33 . hal-01405605

HAL Id: hal-01405605
https://inria.hal.science/hal-01405605

Submitted on 30 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Proposing a Novel Architecture of Script Component to Incorporate the Scripting Language Support in SCADA Systems

Muhammad Waseem Anwar[1], Farooque Azam[2]

Department of Computer Engineering, College of Electrical and Mechanical Engineering, National University of Sciences and Technology, H-12, Islamabad, Pakistan

waseem12@ce.ceme.edu.pk[1], farooq@ceme.nust.edu.pk[2]

Abstract. Scripting language support in SCADA systems significantly enhances their flexibility in performing diverse industry automation functions. Although SCADA vendors do provide rich scripting support in their automation solutions, development and integration aspects of SCADA Script Component are rarely presented in the contemporary research literature. This paper proposes a novel architecture of Script Component in SCADA systems to perform miscellaneous industry automation functions through any scripting language of choice. This architecture is validated by implementing and integrating Script component in a large scale SCADA automation solution. Moreover, series of experiments are performed to validate the working of Script Component after its integration. Empirical results prove that Script Component is capable to perform various SCADA functions through JavaScript by providing enhanced flexibility and simplicity.

Keywords: SCADA, JavaScript, SCADA scripting, industry automation.

1 Introduction

In modern factory and industry automation paradigm, SCADA (Supervisory Control and Data Acquisition) systems are commonly employed. These systems allow efficient monitoring and controlling of industrial processes. Various SCADA vendors provide rich-featured components for automation of miscellaneous industries and factories through single automation solution. However, due to the diversity of industry automation requirements, it becomes difficult to customize the SCADA functions according to particular industry requirements.

Therefore, scripting languages support in SCADA systems are commonly introduced to customize the SCADA functions according to industry automation requirements. Scripting languages support enables the simple and flexible customization of SCADA functions. Hence, it is supported by all renowned SCADA products. For example, GeniDAQ [5] provide scripting support as *Script Designer* and GENESIS32 [4] provide scripting facility as *ScriptWorX32* with rich VB script support.

adfa, p. 1, 2011.
© Springer-Verlag Berlin Heidelberg 2014

Although SCADA vendors do provide rich scripting support, this increases the overall complexity of SCADA products especially for system integrators. The primary reason is syntax and programmatic semantics of scripting languages. Furthermore, complex GUI for script development and lack of proper help documentation makes it more complicated. Moreover, development and integration aspects of scripting languages support in SCADA systems are rarely presented in the contemporary research literature. These factors lead to limited utilization of scripting languages within SCADA systems regardless of their very powerful features.

Therefore, in this paper, a novel architecture of Script Component is proposed to incorporate any scripting language of choice in SCADA systems. Thereafter, proposed architecture is validated by developing and integrating Script Component in SCADA automation solution OpenControl [11]. Empirical results prove that Script Component is capable of executing various SCADA functions with enhanced flexibility and simplicity. Various integration aspects of Script Component are also investigated for security and flexibility of SCADA systems.

This paper is further organized as: Section **2** provides state of the art review. Section **3** provides details of major SCADA components. Architecture of Script Component is proposed in Section **4**. Key benefits of proposed architecture are highlighted in Section **5**. Implementation and integration details of Script Component are described in Section **6** and Section **7** respectively. Validation of Script Component is performed in Section **8**. Future work and Conclusion are presented in Section **9** and Section **10** respectively.

2 State of the Art Review

As the primary idea of this study is to customize various SCADA operations through any scripting language of choice, therefore, it is necessary to first identify the major SCADA software components. Hence, we investigate the renowned SCADA products in order to identify major SCADA components. This comprises Invensys Wonderware [1], Siemens WinCC V7.2 [2], National Instruments LabVIEW [3], ICONICS GENESIS32 Automation Suite [4], Advantech GeniDAQ [5]. We also consider various scientific research works for the identification of major SCADA components. For example, Phan and Truong [17] identified different objects and components of SCADA software. The details of major SCADA components are present in Section **3**.

We further analyze the scripting language support in various SCADA products before proposing our Script Component architecture. For example, GeniDAQ [5] provide scripting support as *Script Designer* with user-friendly interface. GENESIS32 [4] provide scripting facility as *ScriptWorX32* with rich VB script support. Wonderware [1] offer powerful event driven scripting module as *QuickScripts*. We also consider scientific research work relevant to current research context. For example, Marciniak et.al [15] elaborates the concept and advantages of practical usage of scripting languages in SCADA systems by performing experiments on *iFix* Software using VB Script.

3 SCADA Components

The major SCADA components are identified by investigating well-known SCADA products. This comprises Invensys Wonderware [1], Siemens WinCC V7.2 [2], National Instruments LabVIEW [3], ICONICS GENESIS32 Automation Suite [4], Advantech GeniDAQ [5]. The major SCADA components are depicted in **Fig. 1**. Here broad overview of components is presented.

- **Runtime** is the core SCADA component which is responsible to fetch the live values of sensors from PLCs (Programmable Logical Controllers) and other hardware devices. Various implementations of OPC (OLE for Process Control) technology [6] are widely used in Runtime along with different SCADA protocols.
- ***HMI Screen Designer*** component is used for development of HMI screens for monitoring and controlling operations according to real time values of sensors.
- ***Alarm*** component is used to configure and execute desired alarming conditions on live values of sensors. OPC A&E (Alarms &Events) specifications [7] are commonly used in this component however latest OPC UA (Unified Architecture) [8] combine OPC DA (Data Access) [9], OPC A&E and OPC HDA (Historical Data Access) [10] specifications.
- ***Data Logging*** component is used to store live values of sensors into database to manage enormous data logging requirements of critical industrial processes.
- ***Script*** component is used to perform various SCADA functions through scripting languages. VB and Java scripting languages are most commonly used in this component.
- ***Historical Display*** component is used to display historical data in various formats to support management decision making process.

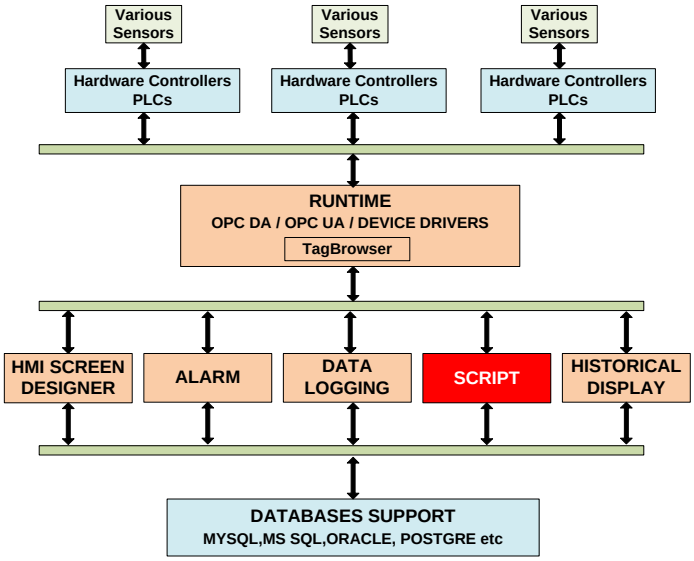

Fig. 1. Major SCADA Components

4 Architecture of SCRIPT Component

SCADA systems are usually developed on OPC technology which is based on Client-Server architecture. Moreover, SCADA systems are commonly used to automate industries where processes are physically distributed over great distances. Hence, Client-Server based architecture of Script Component is proposed as shown in **Fig. 2**.

SCADA applications are normally developed and deployed in two modes i.e. *Designer* mode and *Runtime* mode. In Designer mode, different designing and configuration operations are performed like designing of HMI screens, configuration of alarms, configuration of scripts etc. Once designing and configuration settings are completed, the SCADA application is deployed in Runtime mode where real time operations are performed according to the designing and configuration settings of designer mode. Therefore, Script editor and Script scheduler are only used in Designer mode. The detail of each sub-component is discussed in subsequent sections.

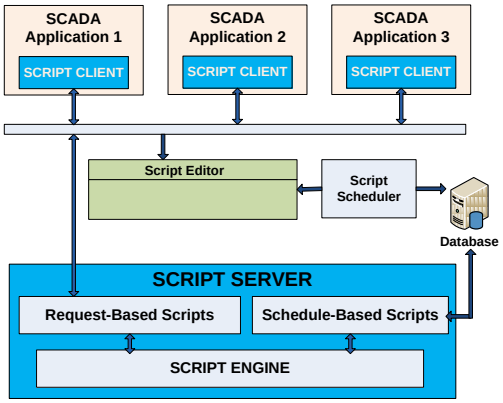

Fig. 2. Client-Server Based Architecture of Script Component

4.1 Script Types

Request-Based Scripts: This type of scripts is always requested by other SCADA components on some special event occurrences. *Typical Usage:* Execution of script on particular event. For example, execution of particular script on Click event of button in HMI screens. *Benefits:* Request-Based scripts provide high level of operational flexibility by executing scripts on special event occurrences.

Schedule-Based Scripts: This type of scripts is executed on the basis of predefined schedule. *Typical Usage:* Execution of script as per configured schedule. For example, creation of database backup after specific time period (e.g. 1 hour) for particular duration (e.g. 1 month). *Benefits:* Scripts are able to be executed as far as Script server is available in contrast to Request-Based scripts where availability of particular SCADA component (e.g. HMI Screen) is mandatory for script execution.

4.2 Script Client

Overview: Script client is responsible to initiate the request of configured scripts to Script sever and return the results back to corresponding SCADA application. *Typical Usage:* Script client provides connectivity with Script server for the execution of Request-Based scripts. *Benefits:* Concurrent processing of multiple scripts requests.

4.3 Script Editor

Overview: Script editor is used for development of scripts according to particular requirements. *Typical Usage:* Script editor provides script development and debugging features in Designer mode. *Benefits:* Script editor accelerates script development process due to its debugging feature, simple interface and inclusion of sample scripts.

4.4 Script Scheduler

Overview: Script scheduler is used to configure the various settings of Schedule-Based scripts. *Typical Usage:* Configuration of Schedule-Based scripts. *Benefits:* Script scheduler provides self-explanatory interface for configuration of scripts.

4.5 Script Server

Script server is the core of Script Component and primarily responsible for execution of scripts. It is further divided into various sub components as shown in **Fig. 3**

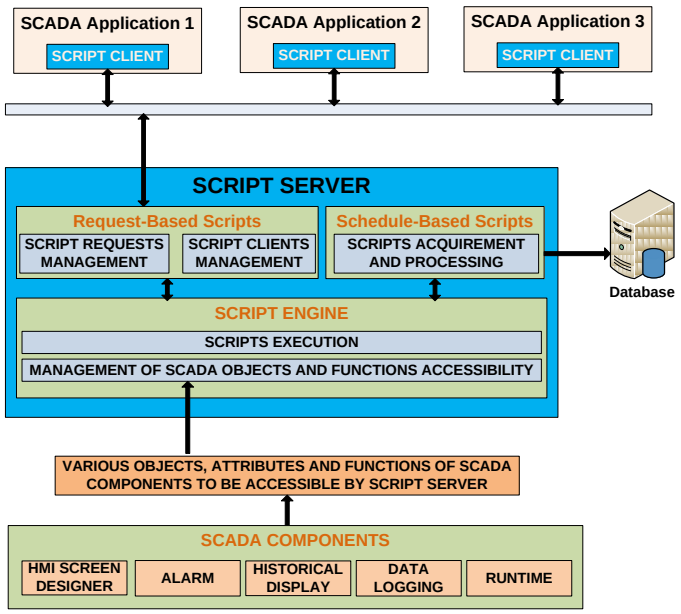

Fig. 3. Functional View of Script Server

Request-Based Scripts Component: This component is responsible to manage Script clients and corresponding script requests for Request-Based scripts as shown in **Fig. 3**. Major functions performed by this component are as follows: -

- *SCRIPT CLIENTS MANAGEMENT:* Request-Based scripts component incorporates the complete mechanism for management of script clients. It provides rich concurrent connectivity features to multiple Script clients. It also maintains list of active and inactive script clients in order to perform different optimization operations.
- *SCRIPT REQUESTS MANAGEMENT*: Request-Based scripts component further maintains the complete list of script requests. The major attributes of this list are Clientid, Requestid and Status. On receiving Script client request, respective Clientid and Requestid are stored in the list and Status is stored as "Processing". Thereafter, requested script is passed to Script Engine for execution which returns the result of script after execution. The result of script is returned to corresponding client by Request-Based scripts component and Status attribute is updated as "Completed" in request list. If there is some error (i.e. connection failure) while sending the result of script, Request-Based scripts component temporarily store the script result and update status attribute as "Pending". It will send the script result on availability of corresponding Script client.

Schedule-Based Scripts Component: This component is responsible to manage different operations for Schedule-Based scripts. The main function of this component is described below: -

- *SCRIPTS ACQUIREMENT AND PROCESSING:* The prime operation of Schedule-Based Scripts component is to obtain all configured scripts from database and investigate the configured attributes for further processing. Script Scheduler (Section **4.1**) is used to configure and save Schedule-Based scripts into database. Schedule-Based Scripts component obtain all scripts from database and store them in structure. Thereafter, it investigates the configured attributes of each script and process the script to Script Engine for execution according to the configuration.

Script Engine: It is used for actual script execution. The major functions of Script Engine are as follows: -

- *SCRIPTS EXECUTION:* Script engine is responsible for executing both Request-Based and Schedule-Based scripts.
- *MANAGEMENT OF SCADA OBJECTS AND FUNCTIONS ACCESSIBILITY:* The key aspect of incorporating Script Component in SCADA systems is to modify SCADA components objects and functions during the execution of script. To accomplish this functionality in Script Engine, classification of SCADA components objects, attributes and functions is developed. On the basis of this classification, access to various objects and functions of SCADA components is incorporated in

Script Engine. Script syntax and invoking matters of SCADA objects and functions are managed by Script Engine for smooth execution of scripts.

5 Key Benefits of the Proposed Architecture

1. **Cross Platform Development:** Proposed architecture is not targeted for specific development technology i.e. .NET, JAVA etc. Therefore, Script Component can be developed in any development technology.
2. **Rich Support for Integration in SCADA Systems:** Proposed architecture is highly supportive for integration of Script Component into any SCADA system of choice because it is based on major SCADA components and client-server architecture.
3. **Flexibility:** Proposed architecture provides high level of flexibility for both development and practical utilization of Script Component. Cross platform development and component-based design provide flexibility for the development of Script Component. On the other hand, inclusion of Request-Based and Schedule-Based scripts provide flexibility to system integrators for development and execution of scripts according to requirements.
4. **Simplicity:** Development and integration complexity is considerably reduced because of component-based design of Script Component. Furthermore, classification of scripts and sophisticated Script editor with enriched scripting features (e.g. debugging, sample scripts etc) further simplifies the utilization for system integrators.
5. **Customizability:** As proposed architecture is based on components, it has high customizability to be tailored as per development and integration requirements.
6. **Reduced Development Time:** Proposed architecture notably reduces development time of Script Component. The primary reason is component-based design and loose coupling between the components. For example, it is possible to develop different components concurrently i.e. develop Request-Based scripts and Schedule-Based scripts simultaneously because both are loosely coupled.

6 Implementation

Script Component is implemented according to proposed architecture (Section **4**). Furthermore, developed Script Component is integrated in SCADA automation solution (OpenControl) which is based on Java technology. Therefore, Script Component is also developed in Java technology with JavaScript support in order to ease integration process. The implementation details of each Script Component are as follows: -

- ***Script Client:*** JAVA RMI technology [12] is used for client-server implementation.
- ***Script Editor*****:** JAVA *Swing* technology is used to develop sophisticated Script editor. Rich script development and customization features are included in Script editor. Further details and relevant screenshots of Script editor can be viewed at [16].

- ***Script Scheduler*:** JAVA *Swing* technology is used to develop Script scheduler. *JDBC* drivers are used to establish connection with MYSQL database. Various attributes can be configured according to requirements as shown in **Fig. 4**. Only *Enable* scripts are considered by Schedule-Based Script Component for further attributes evaluation and *Disable* scripts will never be processed for execution. *Frequency* describes the execution sequence of script and *start /end date* describes the overall duration for which script is applicable for execution. There is provision to select appropriate days for script execution between start and end date. For example, consider a script that is used to calculate the quantity of daily manufactured products in a factory for six month. For this script, frequency should be set to once a day (i.e. 12 hours etc) and start / end date should be set for six months. However, factory is closed on Saturday and Sunday. Therefore, by excluding Saturday / Sunday, script will not be executed on these days for the whole six month duration.
- ***Script Server*:** Java *RMI* technology is used to implement script client-server. Furthermore, *Java Scripting API* [13] and *Rino JavaScript Engine* [14] are used to incorporate JavaScript support in Script server. Java threads are used to provide maximum resource utilization and concurrent execution of scripts.

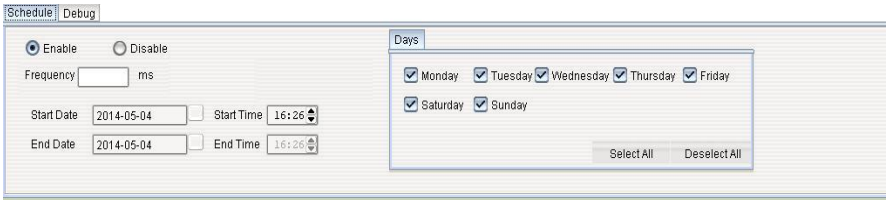

Fig. 4. Script Scheduler

7 Integration

7.1 Request-Based Script Integration

It is important to provide GUI in different SCADA components so that Script editor is invoked for script development on desired event occurrences. Therefore, event-based GUI is provided for the configuration of Request-Based scripts as shown in the **Fig. 5**.

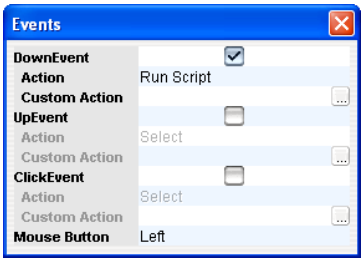

Fig. 5. Event-Based GUI for the Integration of Request-Based Scripts.

Three types of events (Down, Up and Click) are provided for any button type in HMI Screen Designer component. Script execution is allowed through *action* attribute (by selecting run script from drop down list) and Script editor is invoked through *custom action* button.

7.2 Schedule-Based Script Integration

It is simple to integrate Schedule-Based scripts because of their independent execution methodology. Therefore, connection settings of database (MYSQL) are obtained from configuration file and SCRIPT table is created in the database for Schedule-Based scripts storage and retrieval.

7.3 Objects and Functions Accessibility

Security and flexibility aspects are analyzed while providing SCADA objects and functions accessibility in Script Component. *Security:* Integrating Script Component produces critical effects on overall security of SCADA system. Different script functions can be used to produce high security threats in SCADA system. *Flexibility:* On the other hand, limited access of SCADA objects and functions in Script Component significantly reduces its flexibility and strength.

By analyzing the security and flexibility issues, we try to provide reasonable SCADA objects and functions accessibility in order to keep flexibility and strength of Script Component while restricting SCADA security threads as low as possible.

8 Validation

Experimental Setup: Fatek FBS 24MC PLC is used where temperature sensor is attached at analog input and On/Off switch is attached at digital output. Acer Aspire 4720z laptop running Windows XP is used. For simplicity, Script client, Script server and MYSQL database server reside on the same machine. The necessary components (HMI Screen designer and Runtime) of OpenControl are also installed on same machine. As OpenControl is OPC based solution, Buraq Ethernet OPC Server is used for connectivity with PLC. Further details and relevant screenshots can be viewed at [16].

8.1 Request-Based Script Validation

Scenario1: *Turn the switch ON (By modifying the real time switch value) by investigating the temperature sensor real time value*. This scenario provides the model of very common example of real industry automation. For example, in industries, there is common situation where manufacturing plant is start only if temperature of hall is appropriate (e.g. less than 10.52 Celsius). The temperature and switch real time values are fetched via OPC Server and available in Script Component using OpenControl Runtime component. HMI screen is developed where script is executed on click event of button as shown in **Fig. 6**. The executed JavaScript is given below

```
/* including address space of tags */
var tag1="OpenControl.ModbusEthernetDA\\C1.Temperature";
var tag2="OpenControl.ModbusEthernetDA\\C1.ON/OFFSwitch";
/* Request preparation */
OpenGraphTagListener.addTag(tag1);
OpenGraphTagListener.addTag(tag2);
/* Fetching temperature value and Float conversion */
var temperature = parseF-
loat(OpenGraphTagListener.getValue(tag1));
/* Conditional Check */
if ( temperature <= "10.52" )
OpenGraphTagListener.writeValue(tag2,"true");
```

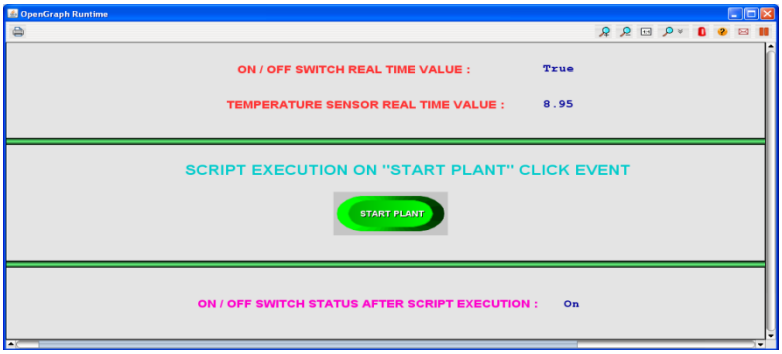

Fig. 6. HMI Screen for Request-Based Script Validation

We perform series of experiments to investigate the Request-Based scripts. Details of experiments, relevant screenshots and videos can be viewed at [16].

8.2 Schedule-Based Scripts Validation

Scenario2: *Adding the real time values of two unique tags /items and write the sum in another tag*. In this scenario, three Read /Write tags are configured in OPC Server through Fatek PLC holding register. Tag1, tag2 and tag3 are used where sum of tag1 and tag2 is stored in tag3 according to configured schedule through Schedule-Based scripts. The value of tag3 can be used by any SCADA component without any interaction with Script Component. This scenario demonstrates the power of Script Component where low level programming functions (i.e. ladder logic) are achieved through scripting.

Further experiments are also performed in order to validate Schedule-Based scripts. More details, relevant screenshots and videos of experiments (including scenario 2) can be viewed at [16].

9 Future work

In this paper, we discuss different features of Script Component to provide scripting language support in SCADA systems. However, certain open issues pertaining to integration and validation of Script Component still need to be researched, as briefly indicated below: -

Enhancement of Integration Capability: we will expand Script Component integration up to all SCADA components because only HMI screen component is integrated in current paper. Furthermore, enrich script calling events will be provided to meet diverse requirements as only few script calling events are included in this paper.

Investigation of Script Component in Distributed Environment: we also intend to investigate the potential response time of Script Component by deploying Script client and server on different machines in distributed SCADA environment.

10 Conclusion

Scripting language support in SCADA systems significantly enhances their flexibility in performing diverse industry automation functions. In this paper, client-server based architecture of Script Component is proposed in order to provide scripting language support in SCADA systems. We introduce Request-Based and Schedule-Based scripts types to perform various SCADA functions by making use of any scripting language of choice. Furthermore, sophisticated Script editor is introduced for the development and customization of scripts. Subsequently, proposed architecture is validated by developing and integrating Script Component in OpenControl. Java technology is used to implement Script Component. Empirical results prove that Script Component is capable of performing powerful SCADA functions through JavaScript.

The key benefits of proposed Script Component architecture are: cross platform development, rich support for integration in SCADA systems, flexibility, simplicity, customizability and reduced development time. This research also provides a strong foundation for *strengthening SCADA systems through scripting languages support*. The researchers in the field of industry automation and SCADA system can significantly benefit from this work as it elaborates various aspects of scripting languages support in SCADA systems.

Acknowledgement

We are thankful to the higher management of Buraq Integrated Solutions to provide source code access of OpenControl for integration and validation of Script Component. Furthermore, rich technical support is always provided by development team during the implementation and integration process of Script Component.

References

1. Inves system, InTouch Wonderware. (2014) http://software.invensys.com/wonderware/
2. Siemens winccv7.2 getting started manual (2013) https://support.automation.siemens.com/WW/llisapi.dll/csfetch/73505596/GettingStarted_en-US.pdf
3. Getting Started with LabView (2013) http://www.ni.com/pdf/manuals/373427c.pdf
4. ICONICS Genesis32 Version 9.3 (2014) http://www.iconics.com/Home/Products/HMI-SCADA-Software-Solutions/GENESIS32.aspx
5. GeniDAQ Automation Software (Accessed 2014) http://www.advantech.gr/products/automation_software/GeniDAQ.htm
6. OPC Foundation (Accessed 2014) http://www.opcfoundation.org/
7. OPC Alarms and Events Specifications (Retrieved 2014) http://www.opcfoundation.org/Downloads.aspx?CM=1&CN=KEY&CI=269&CU=27
8. OPC Unified Architecture (Accessed 2014) https://opcfoundation.org/about/opc-technologies/opc-ua/
9. OPC DA specifications Version 3.0 (Retrieved 2013) http://www.opcfoundation.org/DownloadFile.aspx?CM=3&RI=67&CN=KEY&CI=274&CU=34
10. OPC Historical Data Access Specifications (Retrieved 2013) http://www.opcfoundation.org/Downloads.aspx?CM=1&CN=KEY&CI=276&CU=33
11. OpenControl Open source SCADA automation solution (2010) http://www.buraq.com/products/opencontrol.shtml
12. Java RMI Technology http://docs.oracle.com/javase/7/docs/technotes/guides/rmi/
13. Java Scripting API (Accessed 2013) http://docs.oracle.com/javase/6/docs/technotes/guides/scripting/programmer_guide/
14. Rhino JavaScript Engine (Accessed 2013) https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino
15. Marciniak, P. Kulesza, Z.; Napieralski, Andrzej; Kotas, R," Scripting languages for simulations in modern SCADA systems", MIXDES, pp. 613-618, (2010).
16. SCADA Research Group, College of E&ME, NUST (2013) http://ceme.nust.edu.pk/scadaresearch/scadascript.html
17. Phan Duy Anh and Truong Dinh Chau " Component-based Design for SCADA Architecture" International Journal of Control, Automation, and Systems, 1141-1147, Springer 2010. DOI: 10.1007/s12555-010-0523-y