Bridging SCADA Systems and GI Systems

Simon Back, Simon B. Kranzer and Thomas J. Heistracher
School of Information Technology and Systems Management
Salzburg University of Applied Sciences
Puch/Salzburg, Austria
Email: Simon.Back@fh-salzburg.ac.at
Email: Simon.Kranzer@fh-salzburg.ac.at
Email: Thomas.Heistracher@fh-salzburg.ac.at

Thomas J. Lampoltshammer
Department of Geoinformatics - Z_GIS
University of Salzburg
Schillerstr. 30, 5020 Salzburg, Austria
Telephone: +43 (0)662 8044 7550
Fax: +43 (0)662 8044 7589
Email: Thomas.Lampoltshammer@sbg.ac.at

***Abstract*—Supervisory Control and Data Acquisition (SCADA) systems are omnipresent in production operation worldwide. Failure detection within these systems remains a challenging task as the level of granularity in terms of location-based identification is low. The systems are able to identify and display failing components, but the situation becomes even more severe in case one of the failing components is not an electronic component but a structural component with a spatial extent such as a pipeline system. In consequence, the process of detection and repair of such components remains a time-consuming and cost-intensive task. To provide a generic solution for this kind of issues, the authors present a conceptualization and a software adapter bridging a SCADA system and a Geographic Information System (GIS). The authors employ international standards from both domains to enable an information exchange between these systems. The resulting autonomous software adapter and the corresponding prototypical implementation thus extends SCADA systems with GIS capabilities.**

I. Introduction

In a wide variety of industrial branches such as manufacturing, mining, leisure or security, telemetry is employed to exchange commands and programs as well as to receive monitoring information from local and remote sites. Supervisory Control and Data Acquisition (SCADA) systems comprise these telemetry-based processes as well as data acquisition from sensors within the industrial facility. Based on the gathered data, analyses are carried out and the results are then visually displayed to enable the associated personnel to execute the required actions in due time [1]. However, sensor data do not only feature time-centric facets, they hold location-centric aspects as well (geo-referenced information), which is commonly handled by Geographic Information Systems (GISs). These systems acquire, store, process, analyze, and visualize geo-referenced data, which are gained from sensors. GISs can be found in various application domains such as water resource management, health applications, or energy management [2]. Technically, SCADA systems and GI systems, with their respective time-centric and location-centric perspectives, are not intervened per se and do operate separately. There are no standardized interfaces between these two domains. However, due to their common data source, namely sensors, various common interests arise. For example, the user of a SCADA system could direct maintenance work in case of a failure detected by the sensors in a pipeline based on the geographic information fetched from a GIS.

In this paper, the authors present the conceptional design and the prototypical implementation of a software interface bridging SCADA systems and GI systems. The suggested interface employs standard protocols from both domains. The software prototype translates between the *Openness, Productivity and Collaboration (OPC) Unified Architecture (UA)* for SCADA systems and the *Open Geospatial Consortium (OGC) Sensor Observation Service (SOS)* for GIS. In the following sections, these two communication models are presented and afterwards the conceptual design of the developed inter-domain connection and the related prototypical implementation are described. The paper closes with a summary of the achieved results and an outlook towards future work.

II. OPC Unified Architecture

OPC UA is an internationally approved standard for machine-to-machine communication protocols and the successor of the still widespread OPC standard. A variety of OPC UA-based SCADA systems is under operation in the field of automation technology. In contrast to OPC, which is based on the Distributed Component Object Model (DCOM) that was designed for the Windows operating system, OPC UA is platform independent [3]. Software Development Kits (SDKs) are available for the programming languages ANSI C, C#, and Java. The OPC UA Foundation is also responsible for drafting and approval of all the OPC specifications and it pursues related dissemination activities [3].

A typical implementation of the OPC Unified Architecture consists of a *server* and a *client*. Both exhibit three software layers: The *OPC UA Stack*, the *OPC UA Client/Server SDK*, and the *OPC UA Client/Server* [4]. The *OPC UA Stack* contains different mappings for the transportation of data. The layer responsible for the encoding of the messages defines the serialization of service parameters for a binary format and an Extensible Markup Language (XML) format. The message security layer ensures the process of securing messages, based on the latest standards for Web services. The network protocol is defined by the transport layer: either the UA Transmission Control Protocol (TCP), the Hypertext Transfer Protocol (HTTP), or the Simple Object Access Protocol (SOAP) are employed. By utilizing Web services, facilities and machines that are separated by a geographically large distance become accessible.

978-1-4799-3459-1/14/$31.00 ©2014 IEEE

The address space of OPC UA is given by nodes and related references. Server-provided full entities, such as variables, objects, and data types, are represented by nodes. Relationships are called references and form the basis for a connection between nodes, which are categorized in different classes such as *variable, object, method, view*, or *data type* [3].
The OPC UA standard specification consists of thirteen parts, which are divided into three categories: *Core Specifications, Access Type Specifications*, and *Utility Specifications*. The *Core Specifications* include the first seven parts and describe the core features and capabilities of OPC UA. The *Access Type Specifications* include parts eight to eleven and determine how to apply the core capabilities to specific models of data access. The following models are possible: Data Access (DA), Alarms & Conditions (A&RC), and Historical Access (HA). The *Service Specifications*, which are representing the third cluster, consist of part twelve and thirteen. These specify among other things the process of discovery and identification of OPC UA servers. The following section describes the relevant principles for the implementation as regards the employed GIS standard.

III. OGC Sensor Observation Service

The Open Geospatial Consortium is dedicated to the development of geospatial standards within the GIS domain [5]. The purpose of this consortium is to push forward geospatial standards in order to support the process of developing new and innovative markets, as well as applications concerning spatial technologies. Up to now, there exist more than 30 established and freely available OGC standards for different geospatial scopes [6]. One of these standards is the OGC SOS, which is part of the OGC Sensor Web Enablement (SWE) initiative. OGC SWE develops specifications for sensors, their positioning and communication [7].
An OGC SOS provides an open interface for a Web service in order to retrieve observations and descriptions for sensors and platforms [7]. In combination with other OGC specifications, the standard SOS offers a broad range of interoperability, such as discovering, connecting and querying of sensors, sensor platforms, and sensor networks. Possible scenarios for such an environment could be real-time-based, archived, or simulated settings. The communication between an SOS client and an SOS server is based on XML. Therefore, the query is platform independent. The specification of SOS consists of *Core Operations*, *Enhanced Operations*, *Transactional Operations*, and *Result Handling*. In addition, there exist spatial filters and extensions for communication purposes. The core functions are specifically relevant for the implementation because they include functions for retrieving meta data and detailed information of sensors and for querying observations of sensors and sensor systems [8]. The upcoming section describes the software-based bridge between SCADA and GIS.

IV. Bridging SCADA Systems and GI Systems

While a GIS groups data around a specific location, in a SCADA system data is mainly related to a precise moment in time. Therefore these domains have different point of views on observational data. When it comes to establishing a connection between these two domains, it is critical to avoid loss of information. The project SCADA::GIS is funded by the Austrian Research Promotion Agency (FFG) and aims at integrating information from SCADA systems in GI systems and vice versa without the risk of loss or change of the information discovered in each domain. In addition, the utilisation of specific features of one domain in another domain can hold various advantages: For instance, the real-time behavior of a SCADA system or the tracking of movements in a GIS.
The following sections present a first solution to these given challenges. Constitutive to these results, the authors suggest a standardized interface between these two domains. The following subsections describe the conceptual design of the proposed domain bridge and the related implementation of a software adapter. Afterwards, the description of the development environment and implementation details of a bidirectional connection between OPC UA and OGC SOS are presented.

A. Conceptual Design

The implementation of a software adapter for bridging the gap between SCADA and GIS via OPC UA and OGC SOS can be achieved in different ways based on different technologies. There exist several limiting aspects, however, which reduce the possibilities and which have to be considered beforehand. The software adapter has to feature the following attributes: i) independence, ii) reusability, iii) extensibility, and iv) Internet capability. The software adapter must be able to connect an OPC UA implementation with an OGC SOS without any intrusive dependencies. Thus, the program must be executable autonomously. In order to guarantee usability for various implementations, the programming code has to address this aspect specifically. Hence, the code is divided into several software blocks, which are self-contained in their functionality and have interfaces for communication with other modules. This encapsulation into individual functional areas also facilitates reusability. The ability to establish a connection via Internet is a 'must have' criterion. In addition, it is important to consider different models for sensor measurement data of both domains, specifically in relation to the exchange of data with information of time and location. To provide this information, the data model of the OPC UA server is extended by four variables: *'EPSG', 'x', 'y', 'z',* and *'Time'*. The first variable is intended to specify the European Petroleum Survey Group Geodesy (EPSG)-code for geo-referenced data. The next three variables specify *latitude, longitude*, and *height* above sea level. The last variable specifies the *Time* for each data set in the desired format. The information model of the OPC UA is complete and sufficient for a prototype implementation.

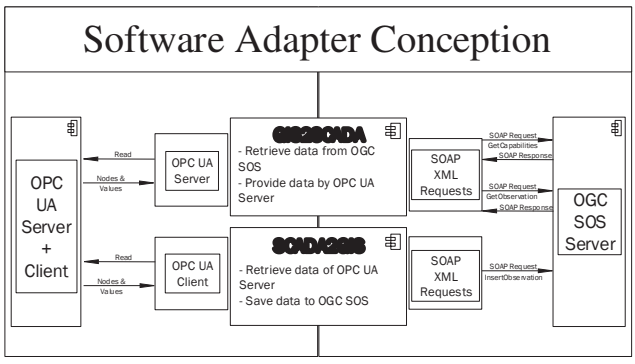

Fig. 1. Software Adapter Conception for Bridging SCADA and GIS

The concept of the software adapter consists of two encapsulated parts that are shown in Fig. 1. The former is called 'GIS2SCADA' and realizes the data transfer from the OGC SOS to the OPC UA server. The latter is called 'SCADA2GIS' and implements the data transfer in the opposite direction. These two parts are executable independently, as this facilitates debugging and speed optimization. Further aspects of Fig. 1 are explained later on in the paper. The next subsection provides information about the development environment.

B. Basic Requirements and Technologies Used

The OPC Foundation merely provides a SDK for OPC UA in form of sample code, which requires a long period of vocational adjustment. In contrast, the company Unified Automation[1] provides a commercial SDK for the programming languages ANSI C, C++, C#, and Java including detailed documentation. In consequence, the present implementation is programmed with the help of the SDK of Unified Automation. For an extended validation of the prototype, the SCADA software Zenon Supervisor[2] from the company *Ing. Punzenberger COPA-DATA GmbH* is used. Due to the fact that OPC is based on the Microsoft Windows operating system and OPC is still installed on millions of computers, Microsoft Windows operating systems are present in many SCADA systems [3]. Hence, for the programming of the software adapter Microsoft technology is employed. Microsoft Windows 7 Enterprise is utilized as operating system and Microsoft Visual Ultimate 2012[3] in combination with .NET framework in version 4.5 is used as development environment. The present software is written in the C# programming language.
Building and processing of XML requests is achieved via the .NET namespace *System.Xml*, which provides classes with functionality for reading and writing XML data. The class *XMLWriter* serves for building XML documents, while the class *XMLDocument* is used for saving XML files. In order to extract specific data from the XML file, the class *XPath* is utilized, which has the same name as the query language *XPath*. This class is specifically designed for fast iteration and selection. The following two subsections describe the implementation of the software adapter for the connection of an OPC UA server and an OGC SOS.

C. Data Transfer from OGC SOS to OPC UA Server

In order to get data from the OGC SOS, two different and correlating SOAP requests called *GetCapabilities* and *GetObservation* have to be sent to the service. This procedure is shown in Fig. 1. These two requests contain well-defined XML data. The received answers can be processed by the namespace *System.Xml.XPath*. *GetCapabilities* requests information about the OGC SOS and its offered operations. The response includes information about queryable offerings, procedures and their respective properties. Additionally, temporal and location-based details are supplied. Based on this information, the XML file for *GetObservation* is created, which requests details about a specific observation. Besides, it is possible to filter the answer by defining temporal intervals and the desired return format. Principal parts of the response are delimited measurement values of the requested observation. Similar to the previous request, information on time and location is also included. In order to provide the requested data for the OPC UA client, an OPC UA server is implemented by the use of the methods of the namespaces *UaBase* and *UaServer* from the SDK provided by Unified Automation. This server is based on a XML data model, which has to be designed before the start and which defines the nodes and references of the server.
After starting the program, a dialog for creating an application certificate is presented, which gathers information about the application name, organization, application Uniform Resource Identifier (URI), and Domain Name System (DNS) name. Towards the successful creation of the certificate, the program presents the connection endpoints. OPC UA offers three security modes: *None*, *Sign*, and *SignAndEncrypt*. If option *None* is selected, the *OpenSecureChannel*, which is responsible for the connection establishment will not be encrypted. Option *Sign* signs the message with the private key of the OPC UA client. The third option, *SignAndEncrypt*, signs the message and encrypts it with the public key of the OPC UA server. After the desired option is selected, the server is ready and is waiting for connections. If an OPC UA client connects to the server, it calls the method *Read* of the implemented server in order to retrieve the measurement data from the OGC SOS. In the following subsection, the data transfer in the opposite direction is described.

D. Data Transfer from OPC UA Server to OGC SOS

In order to provide an easy-to-use client for data retrieval from an OPC UA server by an OPC UA client for saving the information in an OGC SOS, the program features a graphical user interface (GUI), which is partly shown in Fig. 2. Equivalent to the other part of the implementation, the namespaces *UaBase* and *UaServer* of the SDK and their methods are employed to implement the client.

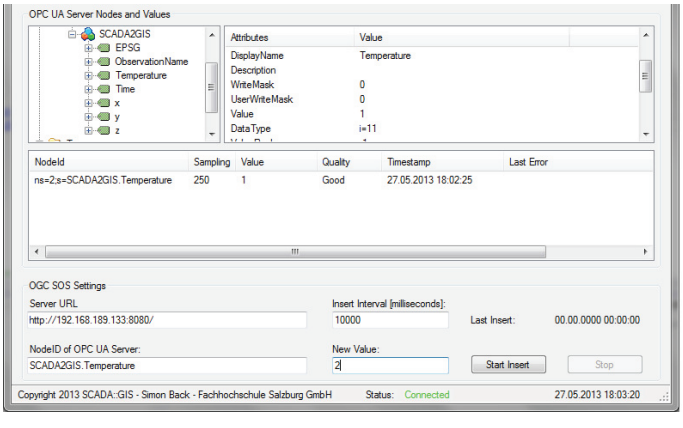

Fig. 2. Part of a Screenshot of the implemented GUI-based application SCADA2GIS

First, the function regarding the discovery of available OPC UA servers in the local network is implemented. Once a server is detected, the respective endpoints and the available security modes are offered. After both options are set, an object of the class *Session* is created, which is passed to the object of *ApplicatonInterface*. Afterwards, two events are registered to

[1]http://www.unified-automation.com/
[2]http://www.copadata.com/de/produkte/zenon-supervisor.html
[3]http://msdn.microsoft.com/library/vstudio/dd831853.aspx

monitor the status of the session and the connection. Then, the method *Connect* is called in order to establish a connection to the specified endpoint's URL. The state of the connection is shown in the status bar in Fig. 2. If successful, the nodes and references of the UA server are presented in a *TreeView* by the methods *Browse* and *BrowseControl*. If the user chooses a variable, it is shown in a *ListView* with its attributes such as *value, name, id* of the node and datatype. The user is now able to drag a desired variable into the field for observations to send its values to the OGC SOS. To transmit data, there exists a changeable, predefined interval of 250 milliseconds. If the user confirms the process by clicking on the button *Start Insert*, a new object of the class *GIS* is created and the method *startInsert* is called. To send the data to the OGC SOS, the XML-based request *InsertObservation* is created by the use of the classes *XMLDocument* and *XmlWriter*. Essential for the request are: i) *time*, ii) value, iii) specification of the sensor, and iv) details of the procedure. Additionally, information about the geographical position can be added. In case the request was successfully sent, the data are processed and saved by the OGC SOS.

At the end of this part of the implementation, the loop shown in Fig. 1 is closed. The next section provides information about the results achieved.

V. Results

Based on the conceptual principles in the fields of automation technology and geoinformatics, an autonomous software adapter was developed as a prototype, whose functionalities were implemented in an encapsulated way to provide extensibility and reusability. The software adapter is based on internationally established standards, namely the OGC Sensor Observation Service and the OPC Unified Architecture. The prototypical adapter provides the exchange of values of measurement observations and corresponding meta data between implementations of these two standards. The present implementation consists of two different and independent applications: A console application, which reads data from an OGC SOS and provides it for an OPC UA server, and a GUI-based application, which reads data from the OPC UA server and sends it to the OGC SOS. The functionality of the prototype was validated by the SCADA software Zenon Supervisor of the company COPA-DATA. The prototype together with its validation presented in this paper proves the successful establishment of a bidirectional connection between a SCADA system and a GI system. Via this connection, measurement data of various information sources, such as sensors, enriched with spatial and temporal information can be exchanged between these two domains.

VI. Future Work

The prototypical implementation of the software adapter presented here leaves some questions open that are relevant for future work. As regards to flexibility, it would be extremely helpful to be able to provide the adapter functionality not in form of a dedicated bilateral link but in form of a distributed service. It can be imagined to employ the adapter service in a cloud environment. By doing so, data could be kept location-independent and could be processed for various purposes that exceed the current approach of near real-time data exchange.

Long term monitoring of measurement data based on data mining techniques is one facet; another facet would be in the field of preventive maintenance or in the field of logistics support for the machine building industry.

As a side effect of the cloud-based approach, a variety of devices such as computers, tablets, and smartphones could be easily incorporated into the present concept. For example, this would enable on-site production forecasting, local operations optimization, and – as a vision – rethinking the way production planning is done today.

Another feature that is planned for the future is some kind of auto-negotiation in order to connect available OPC UA servers and OGC SOSs automatically. This includes discovery of the systems in the network, setting a secure data exchange tunnel including the exchange of security certificates, automatically configuring speed and frequency of transfers, and the flexible definition of temporal conditions. This would result in an autonomously operating software adapter that is capable of auto-negotiation and thus is user-friendly.

Related to these scenarios is the urgent need for dealing with rather complex security issues, which is also a major subject of future work in this field.

Acknowledgement

Financial support for this work was provided by the Austrian research funding association (FFG) under the scope of the 'Bridge program' as project SCADA::GIS (project no.: 834195). The authors would like to specially thank their project partners namely the Research Studios Austria Forschungsgesellschaft mbH (RSA) iSpace, the Ing. Punzenberger COPA-DATA GmbH and the SynerGIS Informationssysteme GmbH. Without the provided funding and the superb research collaboration neither the ideas presented in this paper nor the achieved results would have been possible. Furthermore, special thanks go to Thomas J. Burke (president and executive director of the OPC Foundation) for his comments and valuable feedback in regard to our research.

References

[1] D. Bailey and E. Wright, *Practical SCADA for Industry*. Oxford and Burlington/MA:Newnes, 2003.

[2] M. Gould, M. Craglia, M. F. Goodchild, A. Annoni, G. Camara, W. Kuhn, D. Mark, I. Masser, D. Maguire, S. Liang *et al.*, "Next-generation digital earth: A position paper from the vespucci initiative for the advancement of geographic information science," *International Journal of Spatial Data Infrastructures Research (17250463)*, vol. 3, 2008.

[3] J. Lange, F. Iwanitz, and T. J. Burke, *OPC: Von Data Access bis Unified Architecture*. Berlin and Offenbach:VDE, 2010.

[4] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*. Berlin and Heidelberg: Springer, 2009.

[5] Open Geospatial Consortium, *OGC History (abbreviated)*. [Online]. Available: http://www.opengeospatial.org/ogc/history

[6] Open Geospatial Consortium, *OGC Vision, Mission, & Goals*. [Online]. Available: http://www.opengeospatial.org/ogc/vision

[7] Open Geospatial Consortium, *Why is the OGC involved in Sensor Webs?* [Online]. Available: http://www.opengeospatial.org/domain/swe

[8] *OGC® Sensor Observation Service Interface Standard*, Open Geospatial Consortium (OGC) OpenGIS® Implementation Standard, Rev. OGC 12-006, 2012.