Software Architecture for Automated Fault Analysis: Scalable Deployment and Use of Open Source

T. Popovic[#1], IEEE Senior Member, M. Kezunovic[*2], IEEE Fellow, B. Krstajic[**3], IEEE Member

[#]*XpertPower Associates, College Station, Texas, USA*

[*]*Dept. Electrical and Computer Engineering, Texas A&M University, College Station, Texas, USA*

[**]*Dept. Electrical Engineering, University of Montenegro, Podgorica, Montenegro*

[1]tomo@xpertpower.com, [2]kezunov@ece.tamu.edu, [3]bozok@ac.me

Abstract-- **This paper discusses architecture-significant requirements for an automated fault analysis system. The analysis utilizes substation data collected from event-triggered intelligent electronic devices (IEDs). The proposed architecture assumes generic, transparent, and robust computations that can be applied to variety of IED types and models. When defining a universal solution, the use of standards is critical to facilitating the transparency, scalability, and interoperability. The solution is typically configured to fit the end user needs, which requires configuration traceability, acceptance testing, and change management after the deployment.**

Another aspect of the transparency and scalability of the proposed solution is the use of open source software (OSS), both for the development and deployment of the system. The discussion includes the deployment tools selection and setup. Particularly, the experiences and benefits of using OSS when deploying a system for automated fault analysis are shared.

Index terms – **substation automation, intelligent electronic device, fault analysis, substation data analytics, open source software**

I. Introduction

The data gathered from various intelligent electronic devices (IEDs) installed throughout the power system needs to be utilized in a smart and efficient way. In recent years, the large-scale deployment of IEDs resulted in a massive amount of substation event data that needs be collected, communicated, and processed in a timely fashion [1]. In addition, several new challenges such as cyber-physical security, time-synchronized data storage, configuration management, and efficient visualization need to be addressed as well. The key to efficient use of IED data is to implement fully automated and scalable data analytics solutions. Traditionally, such solutions would be limited to a certain IED type, usually digital fault recorders (DFRs) or digital protective relays (DPRs), which are also vendor and vintage specific. The IED event data was considered non-operational and used in after-the-fact analysis in the past but it is being considered for on-line use now.

This paper discusses the architecture-significant requirements for substation IED data integration and automated analysis solutions. The idea for these solutions is not new and started with first DFRs being installed [2-4]. The requirements are evolving from the experience with previous implementations [5,6]. The presented architecture addresses a broader variety of substation IEDs capable of event-triggered data recording (DFRs and DPRs). The paper illustrates an open architecture that allows transparent approach to the uses of IED data, analytics modules, as well as visualization of the data and results. The proposed architecture is scalable and enables further expansions and interfacing to third-party solutions such as SCADA and satellite maps. A path to achieving transparency of the IED data, analytic functions, and user interface is proper utilization of available standards and non-proprietary formats. Another aspect of the transparency and scalability of the proposed architecture is the use of OSS for the deployment of the solution. The paper addresses experiences and benefits of using open source software both in the development and deployment stages. Approaches for making the future solution scalable, upgradeable, stable and flexible, yet commercially attractive, are discussed with a focus on differentiation of open source benefits and pitfalls. The focus is on the need to understand required guarantees of the code supplier to meet the criteria for standards compliance, testing, and commissioning.

II. Background

The awareness about OSS has been around for some time and its popularity rapidly has risen over the last 10 years [7,8]. This is due to various reasons, but mainly because of the wide availability of the Internet, increased quality of the software offered by the OSS communities, and support from some of the major IT players such as Novell, IBM, Google, and others. OSS offering includes operating systems, databases, networking, virtualization, development tools, and other software that has been widely used throughout industry.

The OSS development promotes evolutionary thinking, and iterative and incremental processes. This is very much in line with modern concepts of software development and agile methods [9]. The traditional sequential development methods processes such as "waterfall" are driven by costs, estimates, requirements defined in advance, and traceability. They are typically not flexible and very vulnerable when a need for a change arises. It is very hard to generalize OSS projects, but they are usually driven by some common interests and communities that form around these interests. The processes are iterative, welcome change, exploration, and there are typically no firm deadlines and detailed requirements specifications set in advance.

The process data diagram of an OSS development is given in Fig. 1. It is a slight update of the diagram originally created by M. Abbing [10]. Typically, the development team is initially created by a group of people that defined the problem. They then form a development team and define initial work plan. It can be defined loosely and refined later as the project grows. The initiation of the project can be initiated by an interest group, various non-profit organizations, governments, and even commercial enterprises. Once the project exists, the process goes through continuous iterations, which include development execution and software release. It is important to understand the roles in the OSS project life cycle. Typical commercial projects include roles such as developers, users, and customers. The customer is a stakeholder financing the project, while the users have to work with the product. The developer is the third role providing the service of the product development. It is not unusual that clashing interests exists between each of these groups [11]. In open source projects there are still roles of users and developers, but the role of customer is distributed between the two. The two groups share interests and users are treated as co-developers [12]. The OSS software is widely used in power industry as well. Examples are operating systems in embedded devices, communication and networking equipment, operating systems for data and application servers, document management systems, virtualization, databases, etc. Even some of the applications come in the form of OSS [13,14].

Fault and disturbance analysis entails taking measurements from IEDs triggered by the fault events and converting them to data, processing data into information, and then using this information to extract knowledge about the fault event [15]. The fault analysis can automatically provide various details, including identification of the affected circuit, whether the disturbance was a fault, fault type, fault location, duration, and evaluation of protection performance. All of this knowledge can be presented to the users and will help them take actions and make decisions more efficiently. This is especially important when there is a need for quick restoration of the system.

Universal and scalable solutions should utilize the available standards to address the data file format, naming, time synchronization, configuration settings, and interoperability. Recordings coming from the IEDs need to be matched with the corresponding IED settings as well as with the correct current power system component parameters. IED-specific settings sometimes come with the IED recordings, but it is not unusual to see those placed in a separate file or even kept on a remote computer. Easy access to the IED settings is critical in order to enable the fault analysis. Some of these issues are being addressed in current IEEE standards development work, including Common Format for Transient Data Exchange (COMTRADE) and Common Format for Event Data Exchange COMFEDE) [16,17]. Another useful standard for handling event-triggered IED data is IEEE file naming convention (COMNAME) [18]. Event-triggered data should be time-stamped and the assumption is those substations IEDs are using time reference from the global positioning system (GPS) of satellites [19]. Handling of the configuration settings can be implemented by interfacing to other systems such as short-circuit program database, relay-setting coordination database, SCADA PI historian, or the International Electrotechnical Commission (IEC) 61850 Substation Configuration Language (SCL) files [20].

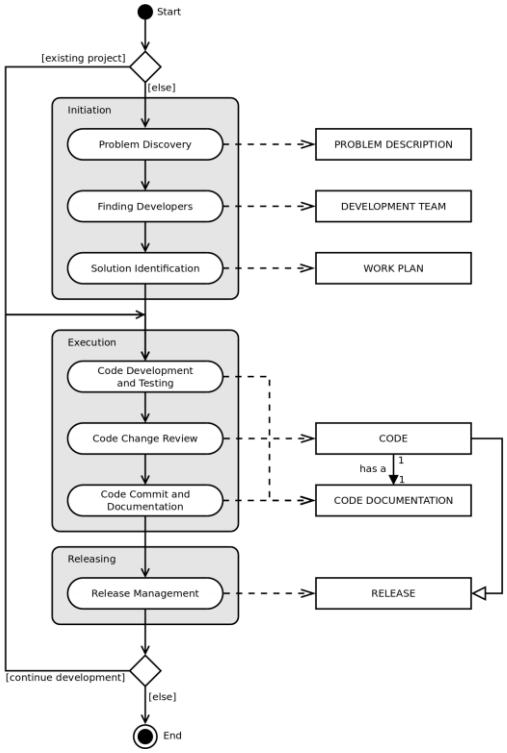

Fig. 1 Open Source Software Development Process

The data analytics solutions can also have their own management and version control for the configuration settings. Exporting results to other solutions may drastically improve the work flow of the engineers who use the solution. Common Interface Model (CIM) and other relevant standards need to be considered [21]. Being open to utilization of the available and future standards is the key to interoperability and scalability of the solution.

III. Implementation Architecture

The implementation architecture for event-triggered data analytics is depicted in Fig. 2. There are two main parts in the framework: a) the data warehouse, and b) interface specifications. The data warehouse contains substation event data, configuration settings, and analytics results (output). Interface specifications define implementation rules for file format conversion (unification), access to the configuration settings, running of data analytics functions, and finally, access to the converted data and analytics reports.

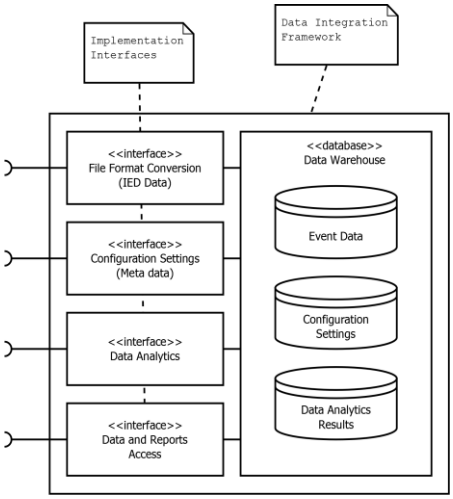

Fig. 2 Architecture framework for event-triggered data analytics

A. Data Warehouse

The data warehouse is the "glue" for the proposed architecture. It is the foundation for the event analysis solution since all of the communication to and from the solution goes through the data warehouse using the rules defined in the implementation interfaces. The data warehouse contains the following types of data:

1) Event Data: all IED-recorded data needs to be converted to selected standard data format such as COMTRADE and COMFEDE. The file repository in the database should utilize a standardized COMNAME file naming convention. It is most likely that the actual file repository integration will require combination of vendor-based and custom developed software modules in order to make sure that the records comply with the selected data format and naming standards.

2) Configuration Settings Data: Besides the IED data, the database has to contain system configuration data that describes: system components and their relationship (i.e. lines, buses, circuit breakers, switches, relays, CTs, VTs, etc.); IED configuration with IED channel assignments and calibration to specific system components (line/bus voltages, line currents, status signals). The system configuration data enables automated IED data conversion into standard formats and integration into the database thus making the data available for new data analytics (software modules). Configuration settings, sometimes called the meta-data, are kept in a readable, non-proprietary formats such as ASCII text and XML [22,23] and, if possible, following the SCL from IEC 61850 [20].

3) Data Analytics Results: The data analytics results should also be kept in non-proprietary and readable formats (ASCII, XML, and even PDF). It is critical to always consider re-usability of the information that is kept in the data warehouse. For the database, it is recommended to use standard SQL command subset supported by various database engines [24].

B. Implementation Interfaces

Four implementation interfaces proposed in Fig. 2 define how each of these functionalities needs to behave, what main functions are required to implement, and what formats to use for the results. Interface concept can be seen as a "contract", which each implementation needs to satisfy. In this framework, the main goal associated with use of these interfaces, is to achieve a universal approach and transparency in data integration, configuration handling, use of data analytics, and presentation of the results. The following sections illustrate the concept using the implementation examples.

1) File Format Conversion: the target is that all of the IED data gets automatically converted into non-proprietary file formats such as COMTRADE and be readily available for further use. Some IEDs do provide tools for exporting data into COMTRADE, or even natively store their records into COMTRADE. However, even then we may need to convert the data since COMTRADE standard has various revisions and allows for lots of freedom with respect to which configuration data is provided or omitted. This can result in files that do provide correct syntax, but are not semantically correct or complete. Real-life examples include situations where the channel units were not correctly assigned (V or A), channel numbers, phase or circuit designations are missing, etc.

2) Configuration Settings Interface: As mentioned before, the configuration settings are needed for the proper file format conversion. The settings are even more critical to proper operation of the data analytics as they provide additional information needed to add the semantics to the measurements stored in IED files. The

configuration tool that provides for entering and editing of the settings related to the file conversion (channel assignments, scaling, and mapping) is needed. Such a tool also provides functions for entering system component descriptions needed for the analysis. For a transmission line, it is used to enter line length, impedance, associated buses, transformers, breakers, and protective relays. All this information is used later by the data analytics or its users. One of the biggest challenges is proper configuration change management. All of the configuration settings can and do change over time. Sometimes, the changes are induced by various upgrades in the system and equipment, but also there are changes of the standards and recommendations that are constantly evolving. Systems for automated substation data integration and analytics are heavily dependent on the settings being correct. All the changes in the settings need to be correctly handled using version control [25].

3) Data Analysis: multiple data analytics functions can be built upon the data integration database and by following the definition of implementation interfaces (Fig. 3). Firstly, the data and report access allows transparency in accessing converted IED data. The data analytics functions do not need to know the details about the IED data source. The same interface is used to feed the analytics reports back to the database. Secondly, the data analytics implement elements of the configuration access in order to retrieve the meta-data needed to interpret the semantics of the IED records. Finally, the data analytics interface provides for a transparency from the system's point of view. Invoking and controlling the data analytics functions by the system or users should be transparent regardless of their inner differences and functionalities.

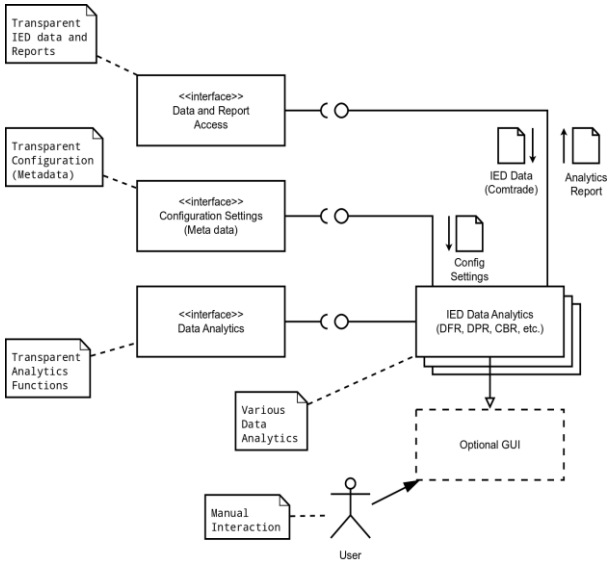

Fig. 3 Implementing the fault data analysis

4) Data and Report Access: the same transparent approach is applied to the implementation of universal graphical user interface (GUI) and dissemination of the analytics reports. The data and report access interface can be implemented in a web-based and desktop-based GUI. Same GUI options for viewing substation IED waveforms and analytics reports can be used regardless of the data source.

IV. Deployment Using Open Source Software

This section describes the OSS tools selection and provides a deployment example.

A. Selection of OSS Deployment Tools

The following OSS tools are used to implement and deploy the proposed architecture:

1) Core technology (Java): the platform selected for developing software is Java [26]. There are several Java development kit (JDK) implementations and Sun (acquired by Oracle) has made JDK available as OSS under the name OpenJDK. Important thing about Java is that it is widely accepted by universities and industry around the globe. Java is platform-independent and it is supported by variety of operating systems. There is a variety of OSS libraries that make life of Java developers easier. Finally, there is a great selection of software development tools for Java coming from OSS world [27].

2) *Operating system:* Linux operating system can be very effectively used for deploying the solutions implemented in Java technology. In this example we used Ubuntu Linux distribution [28]. Ubuntu is easy to obtain and install both in a form of server or workstation. Besides Linux, other operating systems such as BSD (also OSS) and Windows that is coming from closed source commercial world.

3) Application Server: Apache Tomcat is used as an application server: the server side of the solution [29]. It is an open source servlet container and implements the Java Servlet and JavaServerPages (JSP) specifications from Oracle (formerly Sun Microsystems). It also provides a "pure Java" HTTP web server for Java code to run.

4) HTTP Server: Apache2 is used to host web pages relevant to the solution as well as to control the remote file access to the substation data and analysis results [30]. It provides connection to the web application implemented in Java technology using the Apache Tomcat connector.

5) Database: there are several OSS database engines that can be used for the solution. We opted here for PosgreSQL database engine that is available on different operating systems, easy to install and maintain [31]. The solution's data warehouse is implemented combining the database and file repository. The database provides easy access to the meta-data and descriptions of the events and analysis results. It also provides the references to the files in the file repository.

6) User workstation: the workstations only need a standard web browser and Java run-time engine (JRE). A good example of OSS web browser is widely used Mozilla Firefox [32], but others can be used as well. JRE is needed only in case that particular user wants to run Java Web Start programs, which are the desktop applications that load from the web [26].

B. Deployment Example

The solution for substation data integration and automated analysis can be very successfully deployed using open source software. Fig. 4 is illustrating the deployment of the latest generation of the solution using Linux operating system. The components can be deployed to fit the needs of an actual utility and users: substation only installation, centralized, distributed, or regional. The solution is platform-independent and was successfully deployed on MS Windows as well. The solution utilizes Apache HTTP and Tomcat servers, which are secure, scalable, and easy to maintain. The selected open source deployment tools are widely used and supported by virtualization software [33,35], as well as in the cloud computing environment [35]. Key benefits of the use of open source tools are improved portability, interoperability, scalability, security, and maintainability.

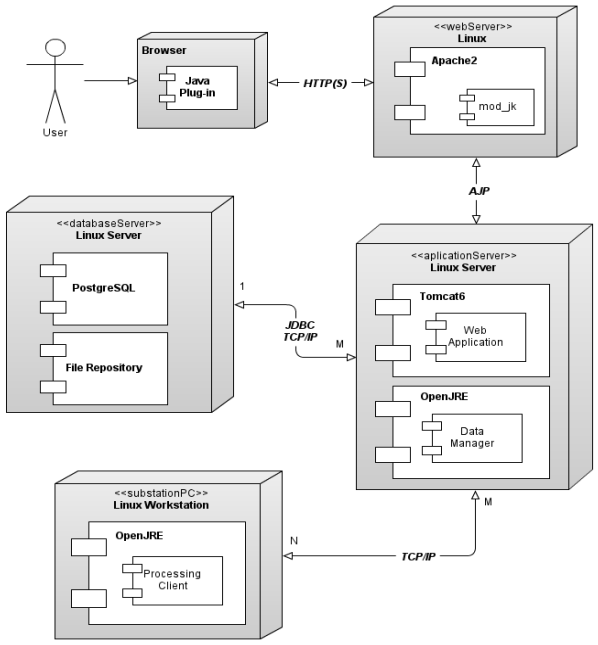

Fig. 4 Automated fault analysis deployment using OSS

The processing client, data manager and database servers are running 24/7 as a service in the background. They are fed the IED event-triggered data through an automated data collection, which provides the incoming folder of the solution with newly recorded event files. The users can access the event data and analysis results using their web browser and the solution's web application. In addition, the users can use a desktop-based report viewer (Fig. 5), which runs on Java Web Start and does not need installation.

The email-based notifications can efficiently be used with desktop, as well as with smartphones and tablets. The analysis results are stored back in the data warehouse and can easily be ported to third-party systems. Examples of such integration with other systems include satellite maps and GIS [36], and SCADA using CIM [15,21]. There is even an OSS Java/CIM library that can be explored and used for such integration [37].

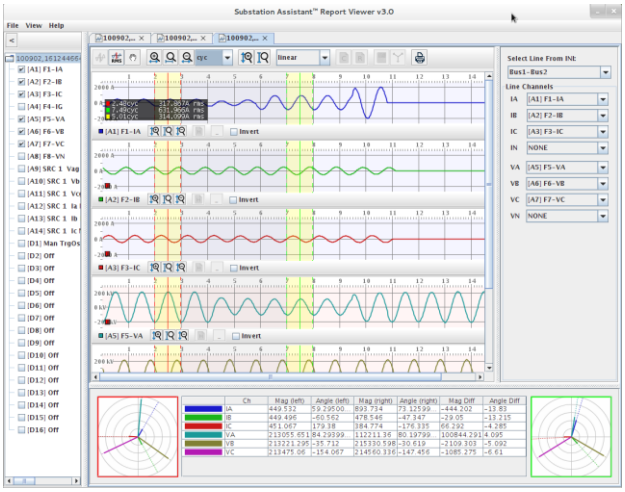

Fig. 5 Desktop-based GUI using Java Web Start

V. Experience and Considerations Regarding OSS

The solution was successfully implemented and deployed using OSS tools, which enabled easier maintenance and interoperability, better scalability, configuration traceability and verification. There are several aspects of OSS adoption and use that include technical, social, legal, and other questions that are beyond the scope of this paper. It is, however, important to acknowledge the quality of OSS offerings such as software development tools, operating systems, databases, and web and application servers. The quality of these packages can be related to the strengths of the corresponding OSS development and management communities and even commercial entities that support these projects.

It remains an interesting topic how to create reliable and sustainable solutions using OSS for automated fault analysis. What would it take to initiate such a project as an OSS and form a community that is capable of successfully growing and maintaining its development remains unanswered. Forming the team could be a challenge as the combination of skills needed is not as commonly available as when dealing with more general type of IT applications. Breaking down the solution into specific algorithms and subsystems before making it an

integrated OSS solution would be desirable. The nature of the application and variations in needs of different users, such as transmission owners/operators or independent system operators, would most likely result in a situation that each user may need a solution configured to fit their specific needs [38]. These specific requirements include the availability of IEDs, communication options, protection schemes, or needs to utilize different fault location calculations [39,40]. In addition, the end users proprietary information and security concerns have to be carefully considered and protected. On the other end, the developers and solution provider's responsibilities, guarantees, and liabilities need to be precisely defined and understood. Risks and benefits associated with open-sourcing the final application product need to be driven by the end user.

VI. CONCLUSIONS

The following is the list of key contributions in this paper:

- The paper discusses architecture-significant requirements for automated analysis of event-triggered fault data collected in transmission substations.
- The proposed architecture aims at universal solution that provides transparent access to substation data and analysis results, which requires proper understanding and utilization of the relevant standards.
- Experience with the use of OSS for development and deployment of the solution is discussed using description of selected OSS tools and the deployment example.
- The paper raises the awareness about the availability and quality of OSS deployment tools and their application in substation data integration and fault analysis.
- The issues and concerns related to opening the source code of the solution itself are addressed. The importance of understanding the impacts of such action is stressed.

ACKNOWLEDGMENT

Authors would like to thank individuals involved in the deployment project over the years:

- Mr. Brian Clowe and Ms. Manjula Datta-Barua with CenterPoint Energy, Houston, Texas;
- Mr. Deepak Maragal with New York Power Authority, White Plains, New York.

REFERENCES

[1] J.D. McDonald, "Substation Automation, IED integration and availability of information," *IEEE Power and Energy Magazine*, vol. 1, no. 2, pp. 22-31, 2003.

[2] A.A. Girgis, M.B. Johns, "A hybrid expert system for faulted section identification, fault type classification and selection of fault location algorithms," *IEEE Trans. Power Delivery*, vol. 4. no. 2, 1989; 978-985

[3] S.D.J. McArthur, A. Dysko, J.R. McDonald, S.C. Bell, R. Mather, S.M. Burt, "The Application of Model Based Reasoning Within Decision Support System for Protection Engineers," *IEEE. Trans. Power Del.*, vol. 11, no. 4, 1996

[4] M. Kezunovic, I. Rikalo, C.W. Fromen, D.R. Sevcik, "Expert System Reasoning Streamlines Disturbance Analysis," *IEEE Comput. Appl. Power*, vol. 7, no. 2, 1994; 15-19

[5] D. R. Sevcik, R. B. Lunsford, M. Kezunovic, Z. Galijasevic, S. Banu, T. Popovic, "Automated Analysis of Fault Records and Dissemination of Event Reports," *GeorgiaTech Fault and Disturbance Analysis Conf.*, Atlanta, Georgia, 2000.

[6] M. Kezunovic, T. Popovic, "Substation Data Integration for Automated Data Analysis Systems," *IEEE PES General Meeting*, Tampla Florida, June 2007.

[7] T. O'Reilly, *Lessons from Open-Source Software Development*, Communications of the ACM, vol. 32, no. 4, pp. 33-37, April 1999.

[8] Open Source Initiative, OSI, [Online]. Available: http://www.opensource.org

[9] M. Cohn, "Succeeding with Agile: Software Development Using Scrum," *Addison-Wesley Professional*, 1 ed., Nov. 2009.

[10] *Open source software development method*, Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Open_source_software_development_method

[11] S. Prehn, "Open Source Software Development Process," *Term Paper, TU Kaiserslautern AG Software Engineering Seminar*, July 2007.

[12] E. S. Raymond, "The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary," *O'Reilly Media*, Rev. Ed., Jan 2001.

[13] IEEE Task Force on Open Source Software for Power Systems, IEEE [Online]. Available: http://ewh.ieee.org/cmte/psace/CAMS_taskforce/

[14] *Grid Protection Alliance products*, GPA, [Online]. Available: http://www.gridprotectionalliance.org

[15] T. Popovic, M. Kezunovic, "Measures of value: data analytics for automated fault analysis," *IEEE Power and Energy Magazine*, vol.10, no. 5, pp. 58-69, 2012

[16] *IEEE Standard Common Format for Transient Data Exchange (COMTRADE)*, IEEE Std. C37.111-1999, 1999.

[17] *IEEE Standard for Common Format for Event Data Exchange (COMFEDE) for Power Systems*, IEEE Standard C37.239-2010, 2010.

[18] *IEEE Standard for Common Format for Naming Time Sequence Data Files (COMNAME)*, IEEE Std. C37.232-2011, 2011.

[19] W. Lewandowski, J. Azoubib, W.J. Klepczynski, "GPS: Primary Tool for Time Transfer," *Proceedings of the IEEE*, 87(1), pp. 163-172, 1999.

[20] *Communication Networks and Systems in Substations*, IEC Std. 61850, International Electrotechnical Commission, [Online]. Available: http://www.iec.ch

[21] *Common Interface Model (CIM), IEC 61970-301*, International Electrotechnical Commission, 2002

[22] *RFC 20: ASCII format for Network Interchange*, ANSI X3.401968, October 1969.

[23] Extensible Markup Language (XML), [Online]. Available: www.w3.org/TR/xml/

[24] *Database Language SQL*, ISO/IEC 9075, [Online]. Available: http://www.iso.org

[25] M. Kezunovic, S. Sternfeld, M. Datta-Barua, D. Maragal, T. Popovic, "Automated Fault and Disturbance Analysis: Understanding the Configuration Challenge," *2011 Georgia Tech Fault and Disturbance Analysis Conf.*, May 2011.

[26] *Java platform*, Oracle, [Online]. Available: http://java.com

[27] J.E. Robbins, "Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools," *ICSE '02*, Orlando, FL, May 2002.

[28] *Ubuntu Linux*, Canonical, [Online]. Available: http://www.ubuntu.com
[29] *Apache Tomcat*, Apache, [Online]. Available: http://tomcat.apache.org
[30] *Apache HTTP Server*, Apache, [Online]. Available: http://httpd.apache.org
[31] *PostgreSQL Database*, [Online]. Available: http://www.postgresql.org
[32] *Mozilla Firefox web browser*, Mozilla, [Online]. Available: http://www.mozilla.org
[33] *Oracle VirtualBox,* [Online]. Available: http://www.virtualbox.org
[34] *VMware Virtualization Software,* [Online]. Available: http://www.vmware.com
[35] *Amazon Elastic Compute Cloud (Amazon EC2)*, Amazon, [Online]. Available: http://aws.amazon.com/ec2/
[36] C. Zheng, Y. Dong, O. Gonen, M. Kezunovic, "Data Integration Used in New Applications and Control Center Visualization Tools," *IEEE/PES General Meeting*, Minneapolis, USA, 2010
[37] A.W. McMorran, "A common information model (CIM) toolkit framework implemented in Java," *IEEE Trans. Power Systems*, vol. 21, no. 1, pp. 194-201, Feb 2006.
[38] M. Kezunovic, S. Sternfeld, M. Datta-Barua, D. Maragal, T. Popovic, "Automated Fault and Disturbance Analysis: Understanding the Configuration Challenge," *GeorgiaTech Fault and Disturbance Conf.*, Atlanta, Georgia, 2011.
[39] M. Kezunovic, "Smart Fault Location for Smart Grids," *IEEE Trans. Smart Grid*, vol. 2, no. 1, 2011; 11-22
[40] L. De Andrade, T. Ponce de Leão. "Impedance-Based Fault Location Analysis for Transmission Lines," *2012 IEEE PES Transmission and Distribution Conf. (T&D)*. 2012; 1-6