INT J COMPUT COMMUN, ISSN 1841-9836
7(5):879-891, December, 2012.

Client Side Internet Technologies in Critical Infrastructure Systems

I. Lendak, N. Ivancevic, S. Vukmirovic, E. Varga, K. Nenadic, A. Erdeljan

Imre Lendak, Srdjan Vukmirovic,
Ervin Varga, Aleksandar Erdeljan, Kosa Nenadic
Faculty of technical sciences
Serbia, 21000 Novi Sad, Trg Dositeja Obradovica, 6
lendak@uns.ac.rs, srdjanvu@uns.ac.rs, evarga@uns.ac.rs,
ftn_erdeljan@uns.ac.rs,

Nikola Ivancevic
EXPRO I.T. Consulting Llc
Serbia, 23300 Kikinda, Svetosavska 43, I/2
ivancevic.nikola@gmail.com

Abstract:
This paper assesses the applicability of client side Internet technologies in software solutions for critical infrastructure systems (CIS). It contains an in-depth analysis of four significant and well known development platforms, namely JavaScript with jQuery, the Google Web Tookit, Microsoft's Silverlight and Adobe's Flash/Flex. They were compared by using the ISO software quality characteristics as comparison criteria. Each of the technologies was applied in a real-life project and the results summarize the authors' experience. The ultimate goal of this research is to enable software engineers to more easily choose a client-side Internet technology when developing a new software solution for the CIS domain.
Keywords: critical infrastructure systems, Rich Internet Applications, comparative analysis.

1 Introduction

Electric power systems, water distribution and telecommunication networks are large-scale critical infrastructure systems (CIS), which can be categorized as safety critical [1, 2]. Their improper use can lead to a discontinuity of critical services (e.g. power outages), or in extreme cases, even to the loss of human lives (e.g. improper safety measures can harm the maintenance crew members). They provide essential services and usually are operated from computerized control centers utilizing various communication systems for data acquisition and control, different software solutions for asset, crew and outage management, and simulation software allowing engineers to plan system extensions, find week spots and optimize the operation of the CIS.
It is a complex task to build software solutions for the aforementioned control center operating a safety critical system, and usually those solutions are highly distributed by their nature. When developing a Web based Human Machine Interface (HMI) for such systems, the system architect can choose from various Internet technologies suitable for production use. Viable technologies exist on both sides of the spectrum: client and server. Relying on a proven technology does speed up and improve development, and it is also a superb tactic for delivering quality into the final product. It comes as no surprise that more and more critical software systems are built by leveraging Internet technologies. Finally, recent tendencies show that even those software intensive systems, which traditionally offered data exchanges solely via some proprietary mechanisms, like a Supervisory Control and Data Acquisition (SCADA) system, are opening up, thus allowing access to their various services through the Internet.
The main motivation behind this work is the authors' aspiration to share with the readers their

extensive experience in incorporating Internet based technologies into software intensive systems built for the electric power and financial domains. That endeavor is still under way, since the transition from a pre-Internet stage to a fully Internet based solution is usually carried out gradually. This transition process is positively influenced by advances in Internet technologies, as new possibilities for solving old problems emerge quite frequently. Thankfully also to the abilities of modern Internet browsers the performance of the contemporary Web based applications is comparable to their desktop counterparts [3, 4].
Modern Internet applications are quite different from the conventional notion of client-server systems, i.e. the client is a lot more than a simple node merely capable of displaying some fully processed static content from the server. Nowadays, the delineation between clients and servers is pretty blurred. Clients are now totally equal with other nodes inside some multi-tiered distributed Web application. This is the principal reason why the authors have chosen to analyze client side Internet technologies in the context of CIS. This paper elaborates some of the recognized client side Internet technologies as development platforms without the intent of showing a definite and complete list of each and every such technology. The analysis is limited to the following technologies:

- Google Web Toolkit (GWT) - an open-source source development kit (SDK) [5] for developing complex cross-platform JavaScript-based front-end applications. The GWT-based applications are written in Java and then compiled into JavaScript by the GWT compiler.
- Javascript/jQuery (JJ) - a scripting language supported by all the latest web browsers. Best suitable for those who require complete control over each line of the client side code. Development is made a lot easier with modern tools, like jQuery [6], a cross-browser scripting library.
- Flex - a user interface (UI) source development kit (SDK) [7] for rich cross-platform Internet applications based on the Adobe Flash platform [8].
- Silverlight (SL) - Microsoft's web technology for Rich Internet Application (RIA) development [9], [10] based on the Microsoft.NET Framework. It offers user interface functionality which is nearing the desktop level.

While in referemce [11] the focus was on a general comparison of client side Internet technologies, this paper performs a similar analysis with CIS in mind, i.e. reflecting on the aspects essential for developing large, reliable and sensitive software intensive systems. The set of characteristics taken into account in each of these technologies were selected from the standard quality model defined in [12]. The following characteristics were selected for this analysis:

- Functionality: interoperability, security
- Reliability: maturity
- Usability: learnability, attractiveness
- Efficiency: time behavior, resource utilization
- Maintainability: stability
- Portability: installability

These characteristics were identified as applicable in the direct comparison of client side Internet technologies when used in large scale critical infrastructure systems. The examination of the

remaining characteristics (e.g. operability, suitability, accuracy, fault tolerance, recoverability, changeability, testability, understandability, adaptability, analyzability) defined by the ISO quality standard was outside the scope of this paper, as they were deemed not entirely applicable for comparing software technologies, i.e. they are more applicable for comparing the software solutions which are developed by using those technologies.
Apart from this introduction and the vital conclusion at the end, this paper is comprised from a detailed discussion of real-life solutions developed with the identified client side Internet technologies, and a comparative analysis of these technologies from the standpoint of industrial systems.

2 Real Life Case Studies

The following three sections contain an in-depth report about each of the chosen Internet technologies when applied in real life project implementations. The first case study describes a modern control center solution for electric power systems [13] developed in both JavaScript and GWT. It is followed by the descriptions of a Meter Data Management (MDM) [14] solution developed in Silverlight, and finally a web based financial system developed in (Adobe) Flex. Each section contains a short description of the project/problem which had to be solved with the application of one of the above listed Internet technologies. These descriptions are followed by short discussions of the applied technologies, focusing on the ISO quality model characteristics listed in the previous section, and aim to analyze the technology itself (e.g. GWT, Flex), instead of the applications developed.

3 DMS Web User Interface

Control center solutions for the operation of truly Smart Grids [16] require varying level of web access. Some customers are content with a web based reporting system available to the management, while others require (nearly) complete SCADA functionality to be available on the Internet and accessible from everyday Internet browsers. While working on various control center projects, the authors learned that in the electric power system domain, web based user interfaces are often required to perform some or all the following operations:

- Visualize geographic diagrams of the grid (or its parts) [17]
- Visualize single-line diagrams of the grid (or its parts) [18]
- Visualize detailed substation single-line diagrams [19]
- Allow easy access to equipment details, e.g. the relevant characteristics of a breaker: nominal current, nominal voltage, maximum breaking current, phases connected, etc.
- Monitor measurement values (obviously not in real-time, but with a certain delay)
- Show simulation function results, e.g. the results of state estimation, load flow [20], [21]
- Allow insight into operations performed by other users, e.g. oversee switching sequences executed by dispatchers

Geographic and single-line views give a high level overview of the complete power system. They provide a means for navigation, zooming and viewing different parts of the network. Detailed substation diagrams allow the users to access information about substation equipment which

*[tbhp]

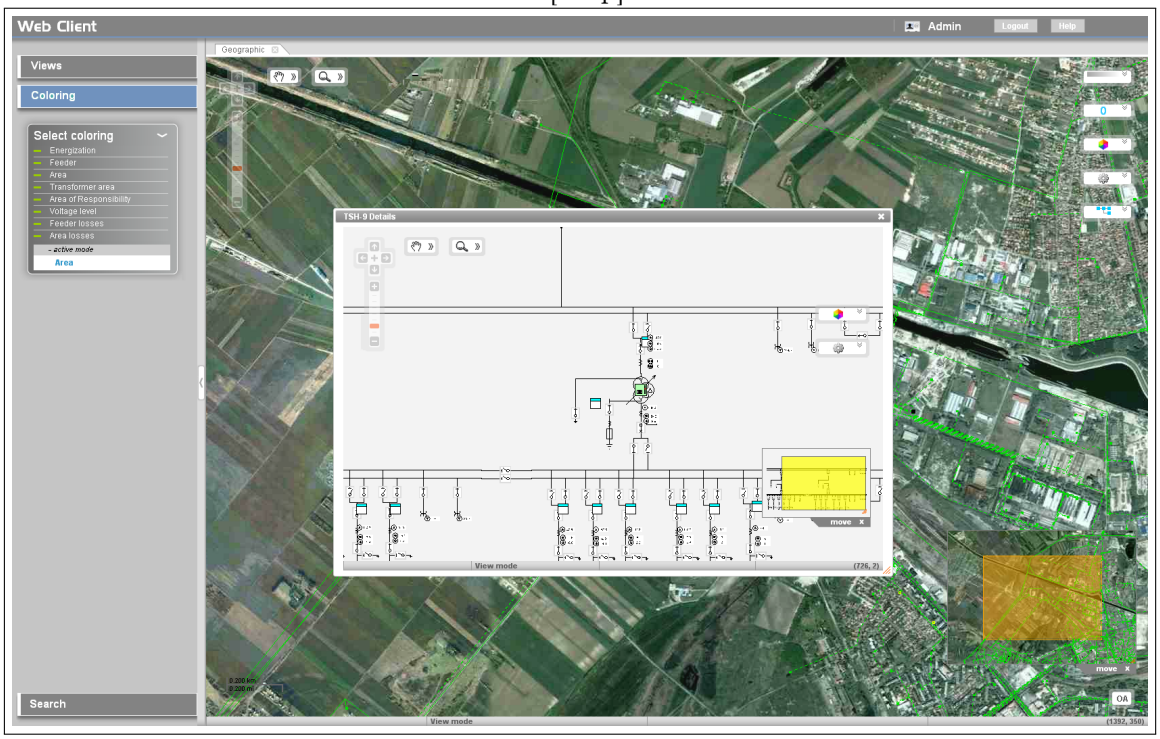

Figure 1: Web view of substation one-line diagram with geographic view in background

is essential in case of outages. Equipment details should be accessible from lists and from the graphical diagrams also. Analytic function results are necessary both online (finding weak spots or overloads not directly measured) and offline (network optimization, planning). Measurement value monitoring allows the user to see these values refresh in (near) real-time.

Fig. 1 shows a snapshot taken from the web user interface developed as part of the Telvent DMS software solution [22]. Apart from presenting the up-to-date state of the electric power system in both geographic (visible in the background in Fig. 1) and one-line diagram views, it also allows insight into the internal structure of substations (as shown in the foreground in Fig. 1) and details of each relevant piece of equipment. Fast navigation in large diagrams is available through the pilot view in the lower right corner. Users can easily navigate to particular network devices using the search tool which allows name and identifier based searches. Besides that, there is a set of handy tools placed on each diagram which improve user experience (e.g. zoom in/out, quick access buttons).

Two versions of this web based user interface were developed: the first technology of choice was the GWT for creating a functional prototype. The second technology of choice was JavaScript/j-Query (JJ). Although quite similar, these two solutions will be discussed separately for clarity. Both are truly thin client applications, fetching diagrams in a graphical format and additional information (e.g. equipment details, names, etc.) in the JavaScript Object Notation (JSON) format [23] from the server side.

3.1 Google Web Toolkit

The Google Web Toolkit (GWT) is an open-source source development kit (SDK) for developing complex cross-platform JavaScript-based front-end applications running in regular web browsers. The GWT-based applications are written in Java and then compiled into JavaScript

by the GWT compiler. GWT was used to build one of the early prototypes of the web based DMS solution, as a kind of a proof of the concept version of the system.
Interoperability was excellent and the GWT proved to be a truly cross platform development platform, allowing the client applications to work on a wide range on destination systems and browsers (Windows, Linux, Apple iOS, Symbian). *Security* was average, e.g. client side code security available through code obfuscation. *Maturity* was moderate in early 2010 and it has considerably improved since then.
Learnability is a matter of knowing or not knowing Java as a programming language and platform, i.e. GWT is straightforward for Java developers and not so accessible for the rest. GWT enables usage of any mature Java tooling system for source code editing, refactoring, testing and debugging. As far as Integrated Development Environment (IDE) support goes, GWT can utilize the well-known Java IDEs. Comparing to a pure JavaScript approach, GWT makes the development process highly efficient. *Attractiveness* is guaranteed by the numerous built-in widgets and libraries which make it possible to implement solutions which resemble the look and feel of desktop applications.
Time behavior on the client side was entirely dependent on JavaScript engine performance, the generated code itself did not have any issues. The GWT compiler optimizes the output code and resource utilization to an extent not necessarily possible when writing JavaScript code manually. The GWT profiler (Speed Tracer) allows developers to fine-tune their GWT applications for the best performance. The execution speed of the JavaScript engine in the latest versions of Internet browsers (Internet Explorer 8, Safari 5, Chrome 19, Firefox 13) allows programmers to develop very complex desktop-like user interfaces with significant computational demands.
Stability was excellent during the complete development cycle of the prototype. *Resource utilization* was considerable at times, slowing down the PC based clients and making it necessary to implement a standalone application for mobile phones instead of running the application in their built-in Internet browsers.
Installability of the solutions built with GWT was excellent and no plug-ins were necessary In order to satisfy one mandatory system requirement from the specification that a full control is necessary over the source code of the application the GWT was (temporarily) abandoned. Although GWT does offer the JavaScript Native Interface (JSNI), which allows developers to write parts of the application in (native) JavaScript, this was not sufficient to satisfy the stringent source code control related system requirement. Nevertheless, by judging the pace with which GWT is improved from version to version, the authors of this web based interface are considering to implement the next version of the system fully in GWT, which might reduce complexity and maintenance costs (compared to JavaScript - see its description below). The stability and quality of the latest version of the GWT obviates the need for such rigorous source code control aspects of the system.

3.2 JavaScript/JQuery

Due to the high level of complexity of the web based Distribution Management System (DMS) [24] graphical user interface (GUI), and because the developers required absolute control over the JavaScript code. Therefore the decision was made to migrate the solution to pure JavaScript/JQuery. This transition was completed in 2010 and the JavaScript version was successfully tested on mobile devices (e.g. mobile phones) just as well as on personal computers.
Interoperability was just as excellent as with the GWT and the JJ solution proved to be a truly cross platform development platform, allowing the client applications to work on a wide range on destination systems and browsers (Windows, Linux, Apple iOS, Symbian). Some limitations to cross-browser support exist and developers in certain cases have to write browser specific

JavaScript code. *Security* was above average, e.g. client side code security is available through code obfuscation.
Maturity of the JavaScript engine and tools was excellent, which came as no surprise as they were around for more than a decade.
Learnability is a major disadvantage of JJ, both because of language limitations and the lack of truly integrated and advanced development environments. *Attractiveness* of JJ as a technology is very limited - people might get put back because of the steep learning curve. Other than that, JJ gives full control and can utilize the rendering capabilities of web browsers to their full extent. *Time behavior* was similar to the one achieved with GWT and entirely dependent on JavaScript engine performance - the full control given to JJ developers might mean a slight advantage over the GWT, as the performance of these applications is entirely under the developer's control. With well optimized Document Object Model (DOM) [25] structures one can implement responsive client applications. It is often better to use the JavaScript Object Notation's (JSON) [23] simple text format for data exchange with the server side as it incurs less parsing overhead and bandwidth use than the eXtensible Markup Language (XML). *Resource utilization* was similar to GWT, i.e. considerable at times, slowing down the PC based clients and making it necessary to implement a standalone application for mobile phones instead of running the application in their built-in Internet browsers. *Stability* was unquestionable during the complete endeavor.
Installability of the JJ based solutions was excellent as no plug-ins were necessary.
At the moment of writing, the latest, JavaScript version of the discussed web based GUI was in its production phase with successful installations worldwide as part of the Telvent DMS software platform [22].

4 Meter Data Management User Interface

Smart metering systems are information systems used in electric power systems which gather measurement values from distant measurement devices (usually 'smart' meters) and integrate them into a complex and unified system for acquisition and control in power distribution systems. These systems are essential components of most Smart Grid [16] initiatives. With the introduction of modern meters it is now possible to control consumption per individual consumers. Meter Data Management (MDM) systems within smart metering infrastructures gather meter data and perform data processing (validation, estimation, editing - VEE).
Large power utilities can have millions of smart meters which periodically report their current consumption rate and receive commands. For the application of algorithms which allow fine grained control of distribution management systems, it is often necessary that these meters send consumption rates as often as possible. This raises significant problems as the millions of meters sending a few measured and status values consume bandwidth, processing and storage resources. Data loss and even compression is not allowed as meter data is used for various mission critical purposes, e.g. billing, load-shedding, etc. The data gathered by smart meters has to be easily accessible to other services within a more complex smart grid (e.g. simulation functions taking into account latest consumption data) as well as to financial people. Therefore, very efficient data processing and storage solutions, a modern user interface and application interfaces built by following the latest industry standards are all necessary components of these systems.
Telvent DMS' Meter Data Management (MDM) system [14] was developed to address the following requirements:

- Search meters and access detailed meter data
- Group meters into groups and areas

- Run validation and estimation functions (where values are not available - linear or spline interpolation is used)
- Display meters on a geographical background
- Data exchange with other software components, e.g. simulation functions, outage management, customer information system, work order management
- Send commands to meters, e.g. on demand read, disconnect, reconnect, ping

The actual implementation of this MDM solution relies on a number of state-of-the-art technologies and tools:

- Meter data is stored in a PI database [26]
- Data access is provided through Microsoft Windows Communication Foundation (WCF) written in Microsoft's C# programming language
- The user interface components were developed in Microsoft Silverlight in combination with Microsoft RIA Services [27]
- Meter visualization on a geographical background is implemented with a specialized component developed by Miner & Miner for their ArcFM (an extension of ESRI's ArcGIS for power utilities) [28]

Fig. 2 shows a SL form from the MDM solution's user interface which displays meter readings with tree view selection and data displayed in both a grid view and a chart. The screenshot was taken during a test session run in the Internet Explorer web browser. The user interface is a thin client running in any web browser with a Silverlight plug-in. The client side consists of approximately fifty forms utilizing advanced components, e.g. grids, charts and the above mentioned geographical background. Drag and drop, direct data editing in grids and deep zoom (i.e. load only visible elements at a certain zoom level) are supported. The thin client was optimized for web based use. The WCF interface is used to exchange custom meter objects which are in line with industry standards [29]. Data security is ensured by data encryption (HTTPs) and user groups with varying access levels.

An MDM form can contain data of up to twenty meters thereby reducing load times. When more than twenty meters should be fetched, paging is used on the client side. The geographical view allows the visualization of up to 3000 meters only on deeper zoom levels.

Interoperability of SL applications is limited to the platforms for which Microsoft issues the latest Silverlight plug-ins. *Security* is inherited from the .NET development platform and the developer has excellent tools for directly influencing security behavior.

As far as *maturity* goes, at the moment of writing Silverlight was becoming a stable and well recognized RIA development platform. The authors' experience was positive, with only a few glitches and memory leaks.

Learnability is excellent due to the use of .NET programming languages which can be easily mastered, the excellence of development tools (i.e. Microsoft Visual Studio) and the high availability of both official learning material and free online resources. *Attractiveness* is excellent and the built applications have a truly desktop-like look and feel. Numerous user interface controls are immediately accessible after installation. .NET version 4.0 comes with an extended set of UI controls which cover most of the needs for business style web applications: grid, chart, tree view control, etc. There is a wide range of open source projects offering additional components, e.g. scheduler, extended charts and grids, etc.

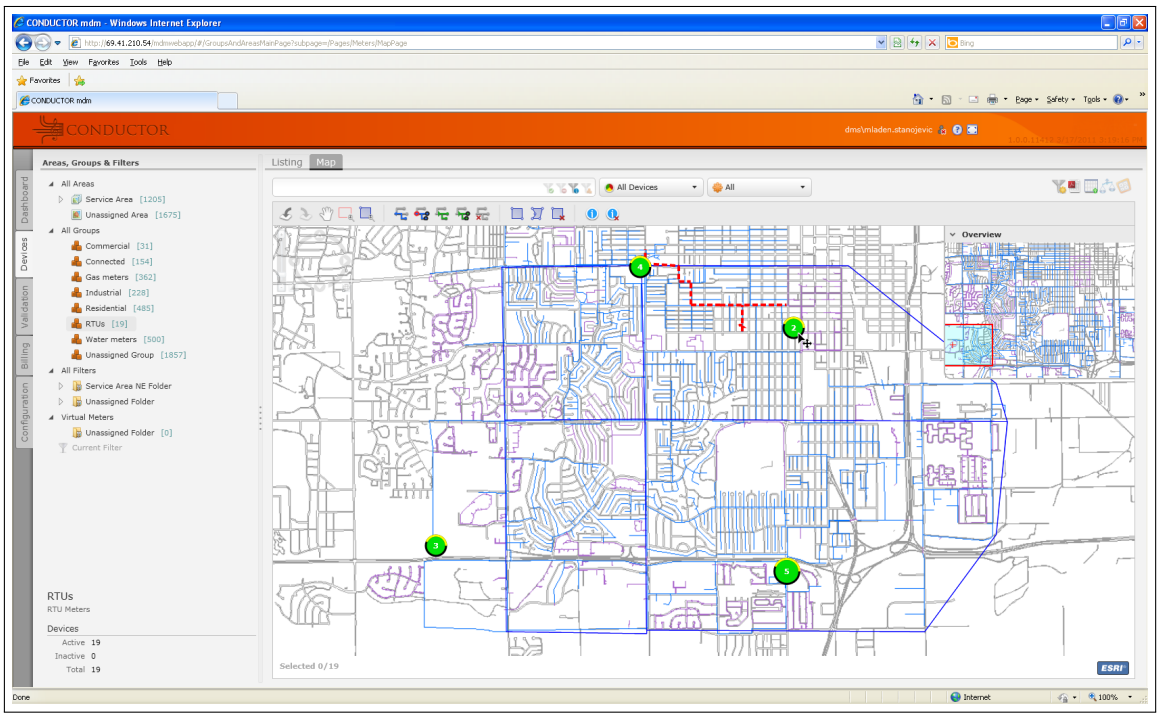

Figure 2: Meter reading data screen developed in Microsoft Silverlight

Time behavior on the client side was above average and quite acceptable even for large numbers of meters with a few optimizations (e.g. paging). *Resource utilization* proved to be moderate which came as no surprise knowing that the code itself is executed by a runtime environment. Network bandwidth utilization is configurable and entirely depends on the developer: the choice of available protocol and serialization formats is wide and can be tailored to specific needs.
Stability was very good on the client side during the complete development lifecycle. There were a few memory leaks.
Installability is somewhat impaired as a plug-in is necessary to run Silverlight based applications. Plug-ins also might not be available for all available hardware and operating system platforms. At the moment of writing the above discussed MDM project was in a release candidate status.

5 Flex in Live Betting

Live, online betting supports the modification of betting instruments (betting markets and prices) during the course of a sporting event. Live betting systems provide the following functionalities:

- management of betting entities (bookmakers, sports, competitions, teams, betting events)
- management of betting instruments (betting markets, prices)
- risk management
- betting events and markets publishing
- accepting players' bets
- betting history management

- processing financial transactions

One such live betting solution is Juliet [30] which the authors of this paper developed in Flash/Flex. It consists of the following major components:

- Core - the component responsible for maintaining the Juliet data model and processing all requests made by the other Juliet components
- BO Console (the back-office) - the back-end application for maintaining the live betting entities (betting events, bookmakers, sports, competitions, etc.)
- Live Console - the back-end application used by bookmakers for creating and maintaining betting events' status, markets and prices
- database (DB) - the Juliet system database
- channel server (CS) - the intermediate component between the UI instances and the rest of the Juliet components
- the user interface (UI) enables players to watch betting events' status, markets and prices and to place bets

The requirements for the Juliet UI were stringent:

- cross-platform application that supports all major browsers
- display information about betting events and markets in a real-time fashion
- easy integration in an existing web site
- possibility to change the look and feel of the UI to match the design of the hosting web site

Fig. 3 shows a screenshot of the Juliet user interface. The left-most block displays the list of betting events (sporting events) available for betting, grouped by the match's sport. The list is automatically updated and contains the betting events which have already started or are about to start in a short time. The status of every betting event in the list (the start time or the current match time, the score, etc.) is shown and updated in real-time. These messages are pushed to the UI from the channel server.
The UI components are constantly connected to the CS and they are receiving notifications as soon as some change occurs. The user interaction is executed completely within the client side (within the hosting HTML page) and the server-side (the CS) is involved only when it is absolutely necessary (e.g. sending a place-bet request). The Juliet UI (see Fig. 3) is implemented as a set of components which can be placed in an HTML page as one or more Flash (SWF) files. This allows an easy integration of the UI components into virtually any website layout the hosting HTML page can have. Also, in order to fit the design of the hosting HTML page, the visual appearance (the look and feel) of the UI components is implemented using a new Flex skinning technique.

The communication layer (the event bus) for exchanging notification events between the UI components and to/from the CS is implemented in JavaScript using the jQuery library.
The Flex experience can be boiled down to the following: the highest level of *interoperability* is ensured by the existence of the Flash player plugin implementations for all major browsers (Internet Explorer, FireFox, Chrome, Safari, Opera) on different destination systems (Windows, Linux, iOS, Android). *Security* of the client side code is excellent as it is stored in compiled

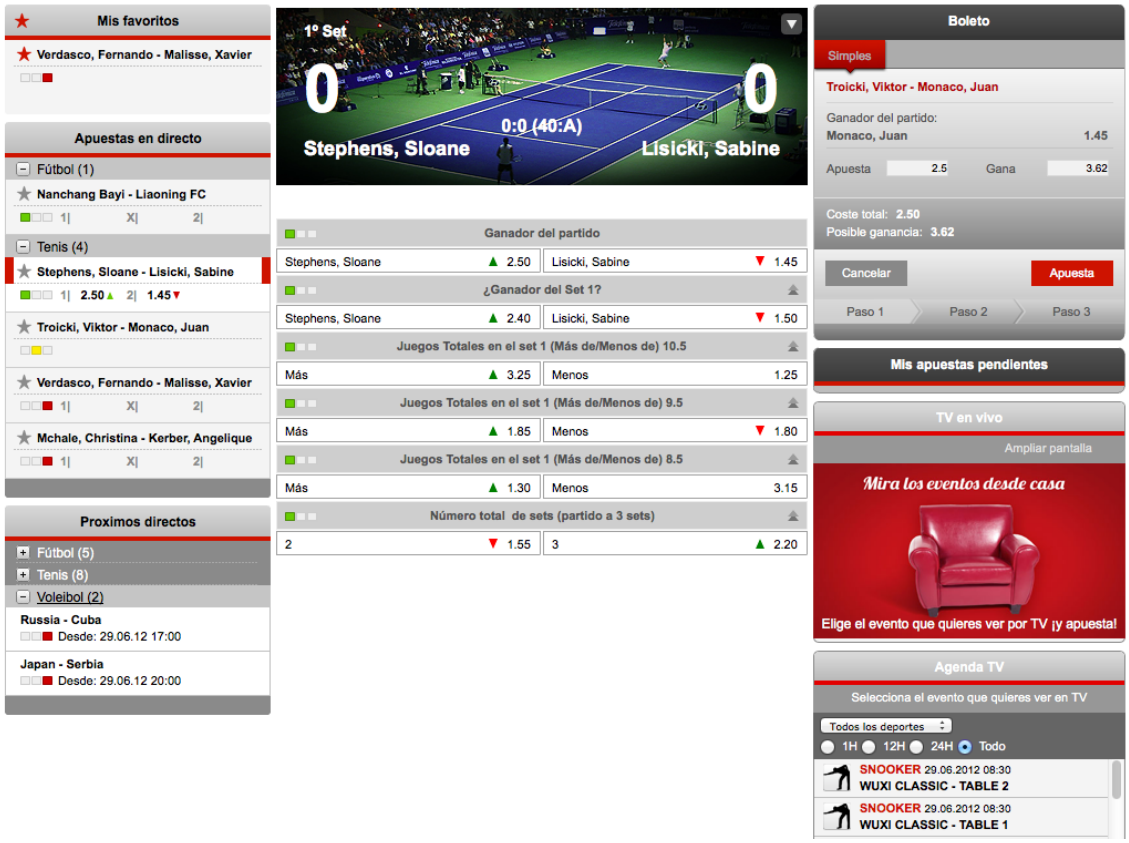

Figure 3: Juliet live betting user interface

meta-code (the byte code for the Flash player platform).
Maturity is excellent as the Flex SDK has been present and constantly improved since 2004.
Learnability is directly affected by mastering new languages that the Flex SDK supports - ActionScript and MXML. In spite of this, the learnability is good because of Adobe's visual tools and excellent knowledge base and documentation. *Attractiveness* is above average as creating visually attractive rich user interfaces quickly, a wide set of server-side integration components and the abundance of third-party widgets makes the Adobe Flex attractive choice.
Time behavior on the client side, once the framework is loaded and initialized, is excellent. The rendering performance of the user inteface is constant and smooth even in cases of great load. The Flex compiler compiles the Flex-based application into SWF byte code while performing possible optimizations. The Flash player itself is able to exploit hardware graphic acceleration to increase the execution speed of graphically intensive application tasks. *Resource utilization* is considerable at the initialization time and load, when the plugin has to download and initialize the Flex framework before the application is ready to start.
Stability is in direct relation to the stability of the Flash player plugin and it was very high even in case of complex demands.
Installability was a bit impaired due to the necessity to install a plug-in. The situation is further aggravated by the fact that some platforms do not have a Flash plug-in, i.e. support for Flash based applications.
The decision to develop this web based, highly sensitive solution in Flex has proven to be correct, as Juliet is online, quite functional with no major issues or downtime during its lifetime.

6 Comparative Analysis

The overview of the client side Internet technologies when utilized in critical systems is shown in Table 1. The table neither contains the definite list of tools for client side application development, nor is the list of applied criteria complete. Each of the analyzed characteristic is assigned a mark between one and five, one being poor and five being excellent. The marks are based on the authors' experience working with the latest versions of the discussed technologies during the complete development cycle of the critical systems discussed in the previous sections of this paper. These marks are a rough indicator of the usability of these tools for developing client side user interfaces in critical systems.

It is visible even after a brief look at the marks in Table 1 that both Silverlight and JavaScript are a 'roller coaster' experience: although they have some excellent characteristics which stand out, they also have serious limitations hampering their immediate use in certain situations, e.g. cross-platform support with Silverlight and learnability and changeability with JavaScript. While Flash has a smoother curve connecting the marks, the smoothest curve and the highest score in general goes to the Google Web Toolkit, which does not excel in all of the analyzed characteristics, but does not disappoint in any of them either.

The marks shown in Table 1 were given based on the experience gained during the development of multiple web based solutions for critical systems. JavaScript/JQuery gives complete control to the developer but needs more time to master then the rest of the tools. Full cross platform support, i.e. operating system and hardware platform independence can be achieved with GWT and JavaScript/JQuery. On the other hand, these two technologies usually incur a bit higher network load, as part of the provided functionality might have to be delivered through bitmap transfer.

Plug-ins are necessary for Silverlight and Flex. These tools themselves and their development environments also require some form of licensing. Application startup is slower as it takes some time while the complete application is loaded into the browser from the server. Silverlight might require a bit higher resource utilization due to the additional overhead introduced by the .NET Common Language Runtime (CLR) running the code.

Each of the above client side technologies has its pros and cons. For very specific and highly specialized user interfaces it might be necessary to choose JavaScript/JQuery or in certain cases GWT, which is only slightly less flexible. On the other end of the range are Flash and Silverlight which are quite modern and allow fast development cycles. Their most notable downsides are the lack of complete cross-platform support and developer freedom.

Table 1: Client side overview (* = poor, ***** = excellent)

Criterion/Technology	Flash	GWT	Silverlight	JavaScript
Interoperability	**	****	*	****
Security	***	***	*****	***
Maturity	***	**	**	****
Learnability	**	**	****	*
Attractiveness	****	***	*****	*
Time behavior	**	****	**	****
Resource utilization	*	***	*	***
Stability	***	***	**	***
Installability	**	*****	*	*****

Applications requiring large amounts of graphical data (see section 3 for an example) might be equally complicated to develop whichever of the shown technology is chosen. Experience gained in the development of such graphically intensive solutions shows that partial, on-demand loading of graphical data (e.g. loading and refreshing only the visible parts of the visualized system) might need custom development with all four presented technologies.

7 Conclusion

The results of this work help mitigate risks in software projects building Internet based critical systems by providing a pragmatic guidance in choosing the right client side Internet technology. The analysis is based on the authors' own practical experience in the domain of critical (infrastructure) systems. A set of ISO standardized quality attributes were used in the assessment of these technologies. They were described through real life case studies from the electric power systems and financial systems domain.

The conclusion is that although modern Internet based client side technologies still have their weaknesses, they are gaining a foothold in critical systems. Applications relying on these technologies can be made fast, secure, reliable and maintainable. The technologies analyzed in this paper will surely become even more widely adopted, as a vehicle in supplying software solutions for critical systems in the near future.

As a future work, the authors plan to extend the list of technologies to be researched, and to create a similar comparison of server side Internet technologies used in the CIS domain.

Bibliography

[1] Karsai, G.; Massacci, F.; Osterweil, L. J.; Schieferdecker, I. (2010); Evolving embedded systems, *IEEE Software*, 43: 34?40.

[2] Parks, R.C.; Rogers, E. (2008); Vulnerability Assessment for Critical Infrastructure Control Systems, *IEEE Secuirty & Privacy*, 6: 37-43.

[3] Fraternali, P.; Rossi, G.; Sánchez-Figueroa, F. (2010), Rich Internet Applications, *IEEE Internet Computing*, 14: 9-12.

[4] Melia, S.; Gómez, J.; Pérez, S.; Díaz, O. (2010); Architectural and Technological Variability in Rich Internet Applications, *IEEE Internet Computing*, 14: 24-32.

[5] Google Web Toolkit; http://code.google.com/webtoolkit; accessed 2011-03-29.

[6] jQuery library; http://jquery.com; 2011-03-29.

[7] Adobe Flex; http://www.adobe.com/products/flex; accessed 2011-03-29.

[8] Adobe Flash; `http://en.wikipedia.org/wiki/Adobe_Flash`; accessed 2011-03-29.

[9] Microsoft Silverlight; http://www.silverlight.net; accessed 2011-03-29.

[10] Pendleton, C. (2010); The World According to Bing, *IEEE Computer Graphics and Applications*, 30: 15-17.

[11] Lammarsch, T. et al (2008); A Comparison of Programming Platforms for Interactive Visualization in Web Browser Based Applications, *12th International Conference Information Visualization*, 194-199.

[12] International Standardization Organization (ISO); ISO/IEC 9126-1:2001 Software engineering - Product quality - Part 1: Quality model.

[13] Bose, A (2010); Smart Transmission Grid Applications and Their Supporting Infrastructure, *IEEE Transactions on Smart Grid*, 1: 11-19.

[14] Vukmirovic, S.; Erdeljan, A; Lendak, I.; Capko, D (2010); A novel software architecture for smart metering systems, *Journal of Scientific & Industrial Research*, 69: 937-941.

[15] Vukmirovic, S.; Erdeljan, A.; Kulic, F.; Lukovic, S. (2010); Software architecture for Smart Metering systems with Virtual Power Plant, *2010 15th IEEE Mediterranean Electrotechnical Conference*, 448 ? 451.

[16] Santacana, E.; Rackliffe, G.; Tang, L.; Feng, X. (2010); Getting Smart, *IEEE Power and Energy Magazine*, 8: 41-48.

[17] Ong, Y.S.; Gooi, H.B.; Chan, C.K. (2000); Algorithms for Automatic Generation of One-line Diagrams, *IEE Proceedings Generation, Transmission and Distribution*, 147: 292 ? 298.

[18] Lendak, I.; Erdeljan, A.; Capko, D.; Vukmirovic, S. (2010); Algorithms in electrical power system one-line diagram creation, *2010 IEEE International Conference on Systems, Man, and Cybernetics*, Istanbul, Turkey, 2867-2873.

[19] Yongli, Z.; Malik, O.P. (2003); Intelligent Automatic Generation of Graphical One-Line Substation Arrangement Diagrams, *IEEE Transactions on Power Delivery*, 18: 729-735.

[20] Cheng, S.; Shirmohammadi, D. (1995); A three phase power flow method for real time distribution system analysis, *IEEE Transactions on Power Systems*, 10: 671?679.

[21] Lendak, I.; Varga, E.; Erdeljan, A.; Gavric, M. (2010), RESTful Access to Power System State Variables, *2010 IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering (SIBIRCON)*, Irkutsk, Russia, 450-454.

[22] Telvent DMS Llc official website; http://www.telventdms.com; accessed 2011-03-28.

[23] JSON.org; Introducing JSON; http://www.json.org/; accessed 2012-06-22.

[24] Popovic, D.; Varga, E.; Perlic, Z. (2007); Extension of the Common Information Model with a Catalog of Topologies, *IEEE Transactions on Power Systems*, 22: 770 ? 777.

[25] World Wide Web Consortium (W3C); Document Object Model (DOM); http://www.w3.org/DOM/; accessed 2012-06-22.

[26] OSI Soft; What is PI; http://www.osisoft.com/software-support/what-is-pi/what_is_pi_.aspx; accessed 2011-03-29.

[27] Microsoft; WCF RIA Services; http://msdn.microsoft.com/en-us/library/ee707344(VS.91).aspx; accessed 2011-03-29.

[28] ESRI; ArcGIS: A complete integrated system; http://www.esri.com/software/arcgis/index.html; accessed 2011-03-29.

[29] IEC (2003); IEC 61968-1: Application integration at electric utilities - System interfaces for distribution management - Part 1: Interface architecture and general requirements.

[30] Juliet Live Betting system; http://www.parspro.com/fp/products/live-betting; accessed 2011-03-21.