Assessment of an Enterprise Energy Service Platform in a Smart Grid City Pilot

Stamatis Karnouskos, Dejan Ilić, and Per Goncalves da Silva
SAP Research, Karlsruhe, Germany
Email: {stamatis.karnouskos, dejan.ilic, per.goncalves.da.silva}@sap.com

***Abstract*—The emergence of the Smart Grid promises a new generation of innovative applications and services that are based on the fine-grained information acquired almost in "real-time" from the underlying infrastructure. To realize this vision, (open) platforms providing access to the smart meter data as well as potential management and value-added functionalities are needed. These will offer basic energy services that can be commonly used by application developers. We depict here our assessment from implementing and operating such a platform in a pilot that took place in Spain in 2012, and draw some lessons learned that affect their design and performance.**

I. Introduction

The emerging Smart Grid [1] implies new roles for all involve stakeholders that now span the value chain in the energy domain. These will increase in number [2] and diversify with the aim to deliver, among other things, a wide-range of better or new value-added applications and services, which may not feasible or make business sense today. Towards this goal, numerous projects investigating multiple facets of the Smart Grid were launched [3]. However, most of the solutions developed start either from the scratch or tackle a significant part of the of the value chain i.e. from smart meter installations, smart meter reading, task-specific data analysis, proprietary applications etc. The later is also a result of the ongoing wars among the stakeholders who try to dominate the area, as well as the non-existence of a common open platform, upon which more sophisticated modular services and applications can be built [4].

Common tasks for all Smart Grid approaches such as analytics on vast amounts of raw data, are not practical if done individually; on the contrary, what is wanted is access to specific results of processing on that data that are highly customized and tailored to application's needs. To this end we expect that platforms, potentially hosted in the cloud, with the capabilities of operating on "Big Data" in a very timely manner will emerge and serve the multitude of envisioned applications and services. With this context in mind, we have designed and operated in a pilot such an open energy services platform i.e. the Enterprise Integration and Energy Management System (IEM) [5] as depicted in Figure 1.

The main aim of our approach was towards enabling lightweight Internet accessible energy services for thin clients over multiple channels, thus lowering the development costs. As seen in Figure 1, there are several architecture parts e.g., the device layer, the middleware, the enterprise services and end-user mash-up applications. In a smart city, numerous

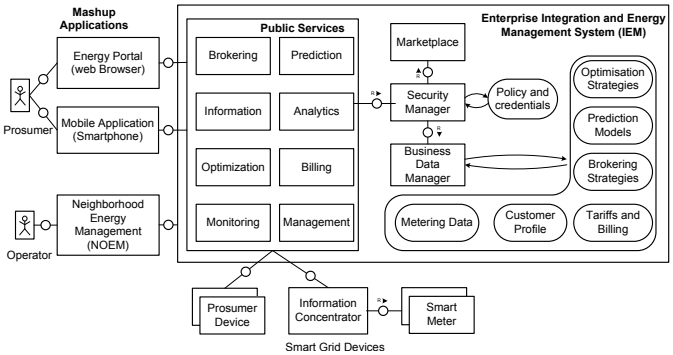

Figure 1. Overview of the IEM architecture [5]

embedded devices may connect directly or indirectly (e.g. via gateways) to the services provided by the IEM. On the IEM service layer, one can mash up services to provide customized functionalities for various applications, such as an energy portal (e.g. accessible via web browsers), mobile applications, or a neighborhood energy management center (NOEM). Furthermore, enterprise services process the collected data and provide advanced functionalities such as validation, analytics, and business context specific processing.

The web services offered by the IEM platform were designed for high performance and use the Representational State Transfer (REST) architecture style, thus simplifying the implementation and integration of thin clients accessing them. The REST adoption imposes some architectural style selections, e.g. client-server separation of concerns, stateless interactions, uniform interfaces and a layered system. As also depicted in Figure 1, several services have been realized in IEM such as: Energy Monitoring, Energy Prediction, Management, Energy Optimization, Billing, Energy Trading (Brokerage) and other value-added services.

IEM (as well as NOEM) have been extensively tested and used operationally in the second half of 2012 as part of the NOBEL project pilot which took part in the city of Alginet in Spain. Data in 15 min resolution of approximately 5000 meters were streamed over the period of several months to the IEM, while the IEM services were making available several functionalities ranging from traditional energy monitoring up to futuristic energy trading. The results presented here, stem directly from the pilot assessment, while in parallel We report in this work some of our experiences during design and implementation, as well as the assessment of the pilot operation of IEM.

II. Data Quality Assessment

One of the key problem areas we were faced with, was that of data quality. High quality data sets are important as they impact all other dependent services such as energy prediction, grid problem identification, energy trading etc. Data may be validated against multiple criteria, e.g. values are expected to be within some limits, check of the syntax, correct time-stamping, duplicate detection, etc. Additional time-stamping at the time of acquisition can be realized in order to enable value-added services such as calculation of metrics such as delays from generation to storage and subsequently integration in the calculation of Key Performance Indicators (KPI).

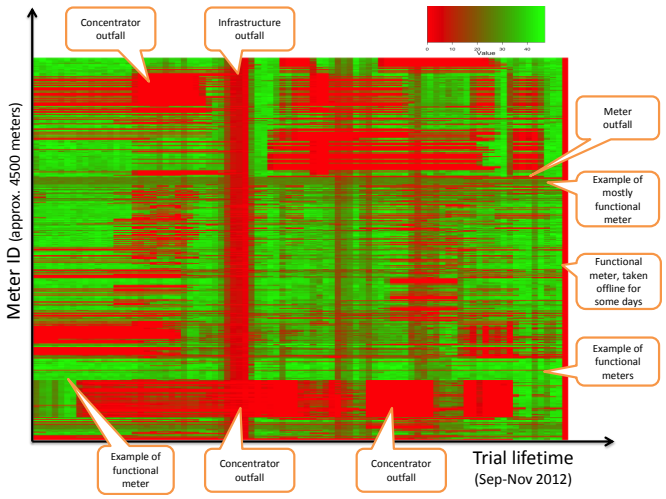

Figure 2. Smart meter readings heatmap for the pilot period

Data processing enables working with the data that is either already stored or is flying in (stream data). This implies data adjustment, e.g. it might be necessary to normalize data, introduce an estimate for a value that is missing, re-order incoming data by adjusting timestamps. Also several calculation functions, e.g. operate on two or more data streams and apply mathematical functions on their composition, as well as transformation are envisioned, e.g. an incoming data stream can be converted on the fly (such as temperature values are converted from °F to °C), or repackaged in another data model etc. However, during the pilot the data quality assessment had the biggest effect on the forecasting of the individual user behavior, especially as this formed the basis for being able to trade electricity online (i.e. buy or sell energy).

In Figure 2 an overview of the three month pilot against the density of data (number of meter readings) received by the IEM is depicted. Some strictly "red" areas indicate problems in the infrastructure, e.g. fallout of a concentrator, delayed or missing data etc. If this heatmap is plot in "real-time" it may assist the energy acquisition stakeholder to identify potential problem areas and initiate response mechanisms to investigate the real reasons, e.g. meter communication problems, infrastructure congestion, malformed data, security problems (such as data replay, reconfiguration) etc. One can also follow the behavior of an individual meter or groups of meters (information concentrator) and their performance in delivering the required data in the expected quality.

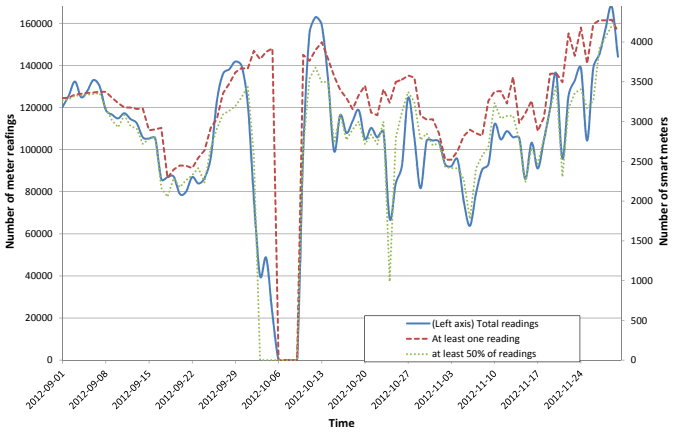

Figure 3. Overview of the accumulated received meter readings per day

Figure 3 depicts a quantitatively representation of the collected data on IEM, analyzed on daily basis. Similarly to Figure 2, events like infrastructure fallout can be identified, but we also get a quick view on the overall smart metering behavior and the load (e.g., number of smart metering events, number of meters reporting measurements etc.) on the IEM side. Being able to map real-world events to the visualized data may provide interesting correlations. For instance, a key event shown is the infrastructure fallout for some days (6-10 Oct 2012), where all meters did not report any data. Although in this case we could trace back the problem in a server failure during a weekend that was followed by a Spanish bank holiday, one can use such metrics to assess multiple aspects such as infrastructure resilience, quality of information provided, etc., that may impact the deployment and operation of future Smart Grid services.

Figure 3 depicts the total count of received meter readings per each day, and we note that the number of received readings follows (as expected) the number of smart meters, indicating that the average number of received meter readings per smart meter, is quite stable. On the right vertical axis the number of smart meters with at least one reading per day is shown, and we realize that even though a high number of smart meters was live (and was expected to deliver meter readings), still an overall low number of them was received. As we see in Figure 3 73% of all smart meters have delivered more than 50% of readings during a day. Still, for some days (excluding the infrastructure fallout), additional analysis on the data revealed that all meters had less than 50% of readings delivered to IEM. This was especially visible before the total infrastructure fallout (which could indicate a warning sign for such events).

As both Figure 2 and Figure 3 depict, being able to assess the quality of acquired data is key into understanding the infrastructure as well as if any future application operation could be supported or what aspects need to be enhanced to do so. In our case, many of the smart meter "failure" to deliver the expected number of meter readings could be traced back to

extensive testing and reconfiguration of the infrastructure and the meters themselves. This had no impact on the real-world, as billing is the only service currently offered live in the city and any subsequent and even delayed meter reading has the accumulated value of energy consumption. However, this had an impact on our pilot services such as energy prediction and indirectly on the trading. It is clear that high quality of data and their timely assessment can provide a much accurate view on what is happening in the grid, and assist with a wide range of value added services.

III. IEM Service Assessment in Pilot

All of the IEM services have been implemented as *Java REST services* deployed in a *Glassfish 3.1 Application Server* (glassfish.java.net) and are accessible over both IPv4 and IPv6. The business data is stored in a *mySQL* DB (www.mysql.com). Specialized analytics and statistics are realized mostly on *R language* (www.r-project.org). All communication with the IEM is done over an encrypted channel i.e. *HTTPS* and a security (with role-based authorization and authentication) framework is in place based on *Apache Shiro* (shiro.apache.org). Additionally for performance reasons, all services interact using *Google Protocol Buffers* (code.google.com/p/protobuf/) which offer a highly efficient binary format. The implementation of the IEM constitutes of approximately 39,000 source lines of code (SLOC) implemented in Java.

A. IEM Service Request Analysis

The IEM services are implemented following the REST paradigm and hence can be accessed via the standard methods GET, POST, PUT, DELETE. Figure 4 depicts all requests made to the available services per group as these are shown in the architecture (Figure 1) The POST method for Monitoring services (or RESTful create) was the most popular, as expected, since smart meter data has been streamed to the monitoring services. Interestingly the Billing service had a lot of POST requests, but further analysis revealed that this was due to the contract creation and their assignment to the customers (configuration for all customers) during the begin of the pilot.

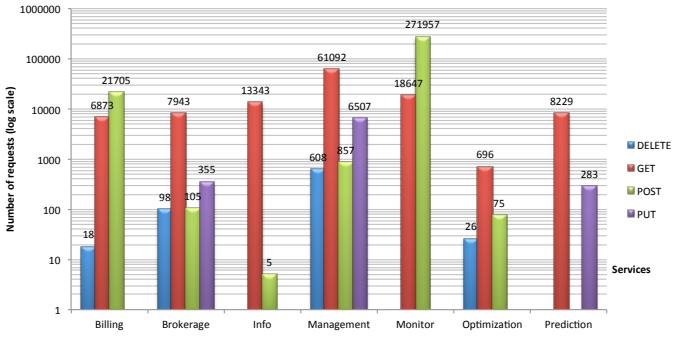

Figure 4. Overview of service categories invocation for different methods

Requests during the pilot were made by three distinctive applications as shown in Figure 1, i.e. an energy portal and a mobile application via which mostly prosumers interacted, as well as the IEM management application (i.e. the NOEM) via which the administrators of the local utility interacted. All the service categories depicted a high number of requests for the GET method. From overall observation of Figure 4 one can conclude that Management, Brokerage, Monitor and Billing services were the most popular ones. Further analysis revealed more detailed info on their exact usage pattern. For instance the increased Management requests can be traced back to the authentication process during the log-in stage of the application(s) etc.

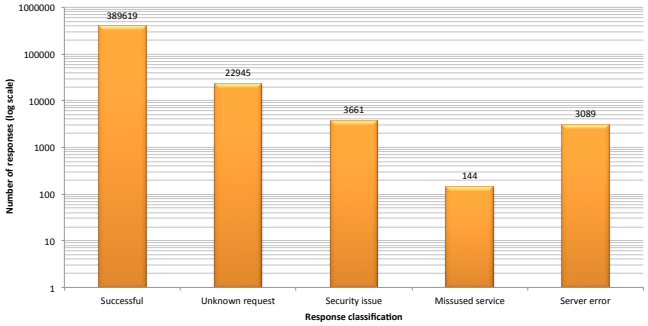

Figure 5. Overview of service response classification

A service invocation as depicted in Figure 1 does not imply that the invocation was always successful. The IEM services return a wide variety of HTTP codes such as success (200,201,204,412), unknown (404,405), security issue (401,403), misuse (400,406,409) and server error (500), so that the service invokers can act accordingly. Figure 5 provides an overview of the classified responses during the pilot, and for instance we can see that more than 92% of successful responses were returned. The "Server error" results were approx. 0.7%, which is surprisingly low considering the complexity of the services and the server load. Of interest was also the results of the "Misused service" classification, which revealed that even for mandatory API adjustments done during the pilot, new functionality could be easily integrated to the applications with minimal effort. We attribute this to the incremental development style as well as the extensive and up to date documentation provided to the developers.

Table I
Overview of mobile device access to IEM

Android Version	Google Play	IEM
2.2 (Froyo)	10.3%	28.27%
2.3.x (Gingerbread)	50.8%	45.88%
3.x (Honeycomb)	1.6%	0.37%
4.0 (Ice Cream Sandwich)	27.5%	17.89%
4.1 (Jelly Bean)	6.7%	7.59%

Figure 1 reveals that one of the end-user applications noted as "Mobile Application" was accessing the IEM. This app was implemented for Android (version 2.2+) smartphones and hence capable of running in a wide variety of mobile devices and tablets. Every call to IEM revealed the nature of the mobile device behind the call (acquired via the HTTP headers). Table I depicts the different versions of android devices identified during the pilot accessing the IEM services, and compares them to the worldwide statistics as these have been measured

in Google Play (developer.android.com/about/dashboards/) in Dec 2012. The Gingerbread version was used to almost half of requests, which is in line to the world-wide statistics obtained by Google Play. The second ranked Froyo version is three times over the Google Play reported distribution, however this may be attributed mostly to the fact that many of the devices available for testing and demonstration were Froyo devices. Interestingly enough, even the latest version of Android is depicted i.e. Jelly Bean, which indicates the existence of some tech-savvy end-users that upgraded their mobile devices such as tablets and mobile phones as soon as the OS version was available (for Jelly Bean July 2012). Such analytics on the data, can provide new insights and help the application developers understand the user base, their devices' capabilities and also obtain indicators about the user himself (e.g. if s/he is a tech savvy one and will try out the latest advanced applications).

B. IEM Server load and DB Analysis

During the pilot, the IEM which provided the services for all applications in NOBEL was hosted in an online server farm (virtual machine) in Germany. The configuration was moderate i.e. a dual-processor multi-core machine, 8 GB RAM and 150GB disk space. The server was reachable only via HTTPS running on port 443, and behind the enterprise firewall. Overall the CPU usage was moderate with some minor exceptions. During the pilot, much of the time the server had been idle, indicating at first that potentially less horsepower would also be enough. However a more intense usage of the IEM services, e.g. by increasing number of simultaneous users or queries operating overwhelmingly on historical data, would increase the CPU requirements significantly. On the memory side however, we witnessed that 8GB of RAM were used almost to the limit, and such multi-faceted functionalities as the IEM provides, especially the ones that do analytics, require more available memory to perform efficiently.

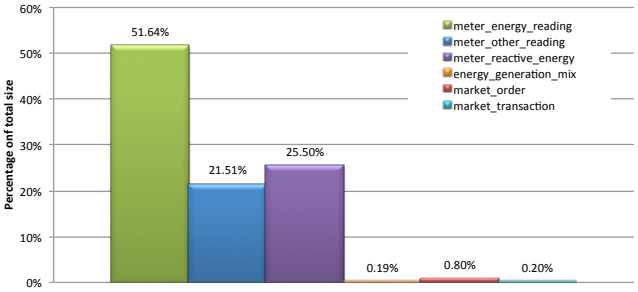

Figure 6. Overview of the six most space consuming DB tables

The IEM server heavily relied on the MySQL database in order to hold all pilot data with a total of 6.1 GB of hard disk space spread over approximately 40 different tables. Since the pilot features more than 5000 consumers (and even more distinct smart meters), the values on Figure 6 are shown in percentage to the total DB size. Interestingly Figure 6 reveals that 98.65% of the space was dedicated to the meter readings i.e. energy reading, levels of reactive energy and other relevant readings (such as Voltage, Power, etc.). Using these percentages one can estimate DB requirements for a future large-scale solution, as well as get a notion where good design decisions are required, e.g space- and performance-wise.

For the SQL queries executed during the pilot i.e. Create, Read, Update and Delete (CRUD) operations, as expected the biggest part is devoted to storage of data, as well as acquiring information from the DB. Since the combined tables for smart meter measurements are responsible for the DB size (as shown in Figure 6), it is not surprising to see that over 53% were SQL INSERT and over 44% were SQL SELECT queries. However, the transmission of data relevant to the queries reveals interesting aspects. Approx. every SELECT query resulted in average to almost 7 times more data than INSERT; more specifically 8.58GB was exchanged in total with an average SELECT of 4880 bytes and an average INSERT weight of 710 bytes.

From the data analysis so far, we can consider that a real-world system implementing the functions offered by IEM should be able to handle increased incoming load while the actual outgoing load depends on the end-user application request rate. However, both incoming and outgoing data rates could be estimated based for instance on the density of data metering or other information acquisition as well as functionality offered at the end-user application side. The communication part does not really offer an insight on the server load, especially when a simple service invocation might result in spikes in the server load due to massive data acquisition and analysis, while the final transferred result may be of minimum size (e.g. a few bytes). Typical example might be the analytics over historic data that spans a custom-defined timeframe of several weeks. Hence, careful design at DB level should consider the expected data flow as well as the service offering and restrictions on their functionalities.

C. IEM Service Performance

The host platform for the IEM services plays a key role on their performance. IEM has been designed to run on a distributed infrastructure and all of its components could (if wished) be installed in different systems with different computational, storage and communication capabilities that correspond to the expected load for those parts. To do so, all components of the architecture depicted in Figure 1 had to communicate strictly over REST APIs and no local dependencies were allowed. We would like to mention again that our hardware configuration was moderate and no real optimization techniques have been applied as mostly default configurations were used. Figure 7 depicts an overview of the response times (in ms) of all services, as well as a categorization of the three applications that were accessing the IEM (as also shown in Figure 1).

Optimization in any aspect may result in better performance for the respective service. As an example, applications located near (network-wise) to the IEM application server resulted in much better response times due the higher network throughput; similarly mobile devices over unstable or low-bandwidth links were performing less efficiently than expected, especially

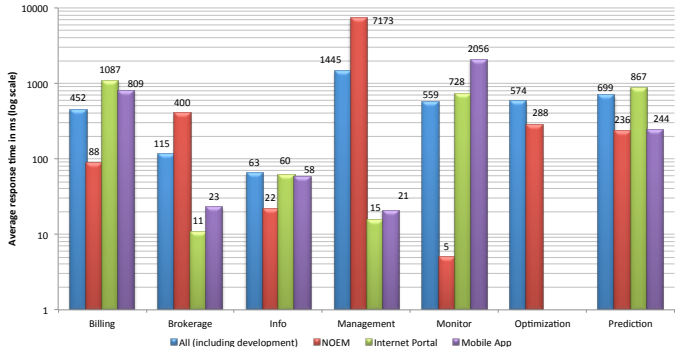

Figure 7. Average response time per requester app for all service categories

when significant amounts of data had to be transferred over the network. The response time is measured from initialization of the request until the reception acknowledgment from the requester's side. Of course, there is also a dependency on how and which services are used and the amount of processing requested on the server side as well as the data to be transferred.

For instance the Management services indicates a high response time for requests coming from NOEM; however this can be fully justified as there are more the 5000 smart meters and more than 5000 customers, which would result in significantly higher payload transfer for Management services called by NOEM (that depict a system view) than the ones called by the other single-user applications where data of a single device/customer were needed. The same logic applies to other services like the Brokerage services. The payload of these two responses may differ more than 5000 fold in our pilot.

The Management services were one of the most frequently used service category. A detailed analysis identified some bottlenecks i.e. when invoking the service for all the customers (average 8235.5 ms) and all the smart meters (average 17737.54 ms), while fetching a single customer or device resulted to 8.36 ms and 7.12 ms respectively. Even though lightweight versions of these services were designed and implemented, the response times are still too high. It is unclear whether the delay is affected by the data fetching time or the composition of the Google Protocol Buffer (GPB) message. Since experiments demonstrated good performance with almost 10000 times bigger tables, the delay might be attributed to the composition of the GPB messages or to the Java Persistence API (JPA). In our implementation there are JPA dependencies of the entities i.e. almost every device has location and information entity attached to it.

An interesting observation can also be made for the performance difference between consumption (average 738.51 ms) and production (average 576.9 ms) services in the monitoring service category. Since both, production and consumption, share the same DB table, it would be expected to have similar response times. However, analyzing the usage patterns of these services during the pilot, revealed that the difference in response time mainly comes from the fact that service consumers were more interested in the consumption readings and they analyzed / visualized them over longer time periods as they wanted to see their monthly or weekly consumption, while production was rarely accessed on such a long time frames (it was rather used only for daily timeslots).

IV. Lessons Learned and Future Work

The performance of various categories of services such as management, brokering etc. during the pilot has been analyzed. However, these are a composite of several other services and an insight on their distinct parts may provide additional understanding. For instance on the Brokerage service we observed that the service fetching the orders of a specific customer (average 217.67 ms) is slower in the response time than the service returning all the orders (average 26 ms). Further analysis showed that the actual usage of the service from the client application introduced these delays (inefficient use of the service). Specifically, the customer order service was not consumed by some of the applications with the best practice guidelines as advised in the developer documentation, but the application developers rather called the generic version of it (without proper parametrization). The last resulted to delivering excessive info i.e. all customer's orders (and then we assume these were filtered on the application side).

Often the developers work under time constraints, and tend to use the simplest way which "just works" without making any considerations on the side-effects such as the server impact or the communication overhead. Although the IEM developers tried to tackle this by being backwards compliant (API-wise) and introduce new optimized functionality, we noticed that many developers of applications simply ignored these developments under the motto of "don't touch a running code", as the effect on the application side (e.g. the mobile phone) was not significant. Hence, in our opinion the default behavior of a service has to be of high-performance and low-load incurring for the server, while additionally offering more advanced options via further parametrization. In that way the majority of the applications connecting will be able to be "migrated" on-the-fly. In parallel strong documentation and a migration period will help.

Scalability is of key importance, especially when considering that all the services now hosted under IEM, will have different usage patterns and performance requirements. We have targeted to make IEM scalable and distributed. Our design decision to enable only RESTful interactions among IEM services and not take advantage of other intra-component calls (which may have resulted in better performance) was justified.

As a multitude of heterogeneous stakeholders are expected to access and use services such as those provided by the IEM, more research needs to be done on the expected usage patterns, data acquisition and validation, as well as the factor that affect "real-time" analytics. Understanding how the infrastructure services are used, helps focusing and optimizing several stages of data lifecycle (from acquisition to processing and end-application-driven adjustment).

For simplicity and maintainability of the platform, it is mandatory that the REST APIs offered, are the same for all

applications. In parallel functionalities identified as "generic" that serve the majority of apps, should be hosted on the server side and include the sophisticated logic. In that line of thought, more lightweight applications can be developed, while their functionalities are decoupled from the data processing logic and intelligence of the service, which can evolve independently. It is expected that in the future such issues will be negotiated by the application developers and the platform providers in order to ensure mutual benefits.

Sometimes the server processing took extensive time (due to the nature of the request or server overload) and in the meantime the client either timed-out or was blocked. Hence, asynchronous behavior was implemented (Request/Acknowledge) instead of synchronous (Request/Response) where possible, in order to avoid the client blocking. This was done during the pilot and has significantly improved the application interactiveness for the end-user. One alternative, yet to be explored, is the use of websockets. This technology, which is available in HTML5, is somewhat between the request/response and publish/subscribe models. The client can initiate a permanent connection to server, effectively giving it a direct route back to the client, and thus enabling it to deliver information when it is available. The impact also of HTTP pipelining as well as new future Internet HTTP-modifying networking protocols like *SPDY* and *HTTP Speed+Mobility* should also be investigated. Additionally, in a real operational environment, it would be expected to make use of high performance in-memory row-based DBs [6] in order to deliver "real-time" analytics over mass data and to very large user bases. As most future devices accessing such services are expected to be mobile, special considerations might need to be taken into account [7].

Security, trust and privacy are challenging issues that are expected to be an integral part of design, implementation and deployment of energy service platforms. In our case, we have not focused on these, but only provided some basic support including authentication and authorization i.e. basic HTTP authentication and authorization were used by all services provided by the platform, and secure interactions over encrypted channels i.e. all REST calls were made over HTTPS. We have placed trust on the end-devices delivering valid data; however device authentication as well as data checks (for replay, modification of values, other sanity checks etc.) should be made in operational environments. Developing secure resilient infrastructures in the Smart Grid era is considered a grant challenge [8].

Apart from the security angle, data quality aspects as indicated here need to be deeply tackled. Missing or delayed data may have a significant impact on key functions such as prediction or analytics, and a cascading effect on decision-relevant processes depending on them, e.g. energy trading, preventive maintenance etc. Hence adequate identification of data quality issues, as well as estimation of missing or delayed values should be further investigated.

Service failures may occur both at client and server side even during the processing of a request. Typical examples are those of network failures, time-outs, service crash, etc. Strategies to detect service performance deterioration and handling are needed. As many of these pose a vivid research area especially in cloud computing domain, we assume that these aspects of service monitoring and lifecycle management will be provided by the underlying platform hosting the energy services.

V. Conclusions

As we can see the design, implementation and development of an energy services platform such as the IEM depicted in this work is a continuous process. Decisions taken on the assumptions prior to pilot had to be revised and adjusted to address real-world aspects including the way services were used. Several aspects could be better understood once the real-users started experimenting with the infrastructure and its services via the provided applications. We consider fundamental the stakeholder agreement of generic energy services that should be provided by platforms and potential standardization [9] of them. Only then these will be able to act as enablers for the creation of lightweight sophisticated end-user applications that can deliver the fully-blown vision of the Smart Grid. This will only be achieved in a collaboration mode among multiple stakeholders and extensive pilots will help fine-tune the proposed solutions.

Acknowledgment

The authors would like to thank for their support the European Commission, and the partners of the EU FP7 projects SmartKYE (www.SmartKYE.eu) and NOBEL (www.ict-nobel.eu) for the fruitful discussions.

References

[1] X. Yu, C. Cecati, T. Dillon, and M. Simões, "The new frontier of smart grids," *Industrial Electronics Magazine, IEEE*, vol. 5, no. 3, pp. 49–63, Sep. 2011.

[2] European Commission, "SmartGrids SRA 2035 – Strategic Research Agenda: Update of the SmartGrids SRA 2007 for the needs by the year 2035," European Technology Platform SmartGrids, European Commission, Tech. Rep., Mar. 2012. [Online]. Available: http://www.smartgrids.eu/documents/sra2035.pdf

[3] V. Giordano, A. Meletiou, C. F. Covrig, A. Mengolini, M. Ardelean, G. Fulli, M. S. Jiménez, and C. Filiou, "Smart Grid projects in Europe: Lessons learned and current developments 2012 update," Joint Research Center of the European Commission, JRC79219, 2013.

[4] S. Karnouskos, "Smart houses in the smart grid and the search for value-added services in the cloud of things era," in *IEEE International Conference on Industrial Technology (ICIT 2013), Cape Town, South Africa*, 25–27 Feb. 2013.

[5] S. Karnouskos, P. Goncalves Da Silva, and D. Ilic, "Energy services for the smart grid city," in *6th IEEE International Conference on Digital Ecosystem Technologies – Complex Environment Engineering (IEEE DEST-CEE), Campione d'Italia, Italy*, Jun. 2012.

[6] F. Färber, S. K. Cha, J. Primsch, C. Bornhövd, S. Sigg, and W. Lehner, "SAP HANA database: data management for modern business applications," *SIGMOD Rec.*, vol. 40, no. 4, pp. 45–51, Jan. 2012.

[7] N. C. Zakas, "The evolution of web development for mobile devices," *Queue*, vol. 11, no. 2, pp. 30:30–30:39, Feb. 2013.

[8] S. Karnouskos, "Cyber-Physical Systems in the SmartGrid," in *IEEE 9th International Conference on Industrial Informatics (INDIN), Lisbon, Portugal*, Jul. 26–29 2011.

[9] J. Bryson and P. D. Gallagher, "NIST framework and roadmap for smart grid interoperability standards, release 2.0," National Institute of Standards and Technology (NIST), Tech. Rep. NIST Special Publication 1108R2, Feb. 2012. [Online]. Available: http://www.nist.gov/smartgrid/upload/NIST_Framework_Release_2-0_corr.pdf