Towards an adaptive SOA-based QoS & Demand-Response Provisioning Architecture for the Smart Grid

Christos Chrysoulas, and Maria Fasli

***Abstract*— Dynamic selection of services and by extension of service providers are vital in today's liberalized market of energy. On the other hand it is equally important for Service Providers to spot the one QoS Module that offers the best QoS level in a given cost. Type of service, response time, throughput, availability and cost, consist a basic set of attributes that should be taken into consideration when building a concrete Grid network. In the proposed QoS architecture Prosumers request services based on the aforementioned set of attributes. The Prosumer requests the service through the QoS Module. It is then the QoS Module that seeks the Service Provider that best fits the needs of the client. The aforementioned approach is well supplemented with a data analytics/machine learning architecture to further enrich the provisioning aspect this work is bringing to the Smart Grid market of energy.**

***Index Terms*—Data Mining, Machine Learning, QoS, Service Oriented Architecture, Smart Grid**

I. INTRODUCTION

IN a constantly growing and demanding market of energy environment, there arises the need for a Quality of Service (QoS) mechanism to properly support the constraints that are imposed by the consumers of energy, without neglecting the importance of keeping the balance of energy flow in the network in an as stable as possible level.

In today's liberalized market of energy playground, it is more crucial than ever to seamlessly provide the end users with the requested services, without putting in jeopardy the grid's stability. In order to properly achieve this goal, an in advance way of placing, scheduling, and assigning the requests for energy consumption (or even for energy production) should be considered. A mechanism with respect to attributes like: type of service to be served, response time, availability, cost and probably throughput should be developed and adopted in order to smoothly pass from the classic energy grid to this new more intelligently build Smart Grid era.

Throughout our study, we try to enforce the Service Oriented Architecture (SOA) approach to the Smart Grid field. That was triggered by noticing that in the Smart Grid field the whole action is initiated by two main actors, namely the Consumer (in our case the Prosumer/User) and the Provider (in our case the Aggregator) of energy (the service). See Fig. 1 for an abstract representation. This is the angle from which the SOA is superintending a system. Based on the aforementioned, we tried to make use of what the SOA field has to offer in order for different Providers to be able to independently create their services and seamlessly "feed" the Consumers. This approach is worth adapting to the Smart Grid environment.

To efficiently deliver energy resources in the smart grid, an energy resource management strategy needs to be developed to balance the energy demand and supply. Developing effective energy resource management schemes is challenging due to numerous fluctuations the entities on both the demand and supply sides experiencing. For example, on the supply side, fluctuations could come from distributed renewable energy resources due to solar irradiance, wind speed, etc. On the demand side, numerous effects, including natural disasters, plug-in vehicles, personal habits of using energy, weather and temperature, etc., could make it difficult to predict energy usage. In this paper, we develop techniques to effectively manage energy resources and usage in order to provide the needed stability to the grid. Particularly, to balance energy demand and supply, we develop effective techniques to accurately model and forecast the amount of energy generation and demand over time.

The rest of the paper is structured as follows. In Section II, the motivation for bringing QoS in SOAs is described. Section III gives a detailed presentation of the proposed QoS approach. Section IV presents the proposed mining approach for the Smart Grid and how it can serve the demand –response of energy. Section V presents how the results can be used for visualization purposes, while Section VI provides the conclusions, and outlines future work.

II. QOS IN SERVICE ORIENTED ARCHITECTURES – RELATED WORK

SOA provides the means for developing software in the form of interoperable services. Providing a common programming

Manuscript received January 15, 2017; revised May 16, 2017. Date of publication: June 1, 2017.

C. Chrysoulas is with the University of Essex, School of Computer Science and Electronic Engineering, and Essex Business School, Wivenhoe Park, Colchester CO4 3SQ, UK. (e-mail: cchrys@essex.ac.uk).

M. Fasli is with the University of Essex, School of Computer Science and Electronic Engineering, Wivenhoe Park, Colchester CO4 3SQ, UK. (e-mail: mfasli@essex.ac.uk).

Digital Object Identifier (DOI): 10.24138/jcomss.v13i2.375

interface, through which any application can be accessed [1] is the added value that the service-oriented development brings to the IT world. So, any service is defined as a discrete unit of functionality which is made available through a service contract [2].

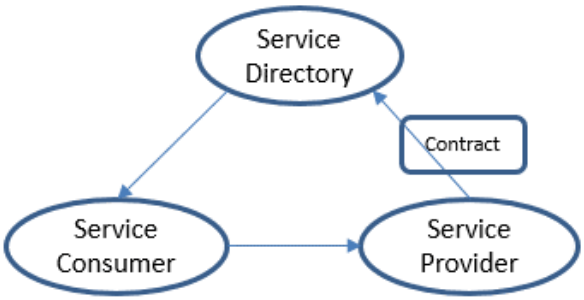

Fig. 1. Service Oriented Architecture Overview

A service contract specifies all interactions amongst the various actors, and are: i) Service interface; ii) Interface documents; iii) Service policies; iv) Quality of service (QoS); and v) Performance.

The fact that a service is explicitly managed is what differentiates a service from other software constructs like components and/or objects. A service level agreement (SLA) is responsible for the management of the QoS and the performance. Additionally, the entire service life cycle is managed (from design, to deployment, to enhancements, to maintenance).

But what is the driving force for adopting SOAs? The need for code and systems re-usability that SOAs offer is the reason for shifting to SOAs [3] rather than using highly specialized building blocks, which most of the times are application specific. A service must come with the following characteristics: i) hide its internal logic; and ii) be loosely coupled, with no predefined connections, but with clearly defined inputs and outputs. SOAs can easily support QoS features and behavior by putting their characteristics in the WSDL description of a requested or provided service. SOAs message exchange is based on XML, so a flourishing in the description is needed to make it possible.

QoS in Grid computing was studied in GARA [5]. In GARA approach, the separation of resource reservation and actual allocation is proposed for supporting critical requests. Studies of Ran [6] and Tian [7] concentrated on extending the first one the UDDI registry and the second one extended the WSDL files in order to bridge the gap between the Web Service layer and the network layer. To our knowledge both approaches lack implementation and validation reports.

Numerous approaches for providing QoS support in middleware based models, and specifically message oriented middleware models can be found in the bibliography. The Quartz [8] approach needs a large dataset (meaning large number of attributes) in order to provide adequate QoS support amongst different application areas. In [9] the QoS negotiation is in advance takes place by communicating a QoS contract amongst the involved parties. Our approach is in position to also send alternative offers to the Prosumers.

Cucinota et al. [10] presented a SOA approach that allows negotiation of the individuals QoS characteristics. In this way any unwanted interference amongst different services can be avoided. In [11], a negotiation architecture was developed where a QoS Manager detects any possible QoS violations, communicates with the resource manager and starts a new negotiation among the interested parts. Our model is proposing the most fitted to the Prosumer's needs QoS offer based on mining techniques and by processing the outcome with the help of machine learning algorithms.

Papazoglou et.al. [15] present an overview of the current research in service oriented systems and how SOAs are aiming to the efficient and automated provision of managed services which particularly during runtime are subject to dynamic and adaptive change processes. The research is in depth analyze what service management really is. The service management not only has to cover the installation, first configuration and monitoring of services but should also be in position to serve the needs of re-configuration and life-cycle management in order to support self-configuration, self-adaptation, and self-healing. In this way the need for service versioning and dependence management can also be achieved.

When it comes to the actual implementation, managing dynamically adaptive service systems implies that the various elements of the service implementations can suitably and efficiently be managed at runtime. To serve this need, many authors propose combinations of service oriented architectures with software component based implementation approaches. Chrysoulas et.al. [16] report on the FlexiNET project which applies a special Grid-oriented component model in order to master dynamic service deployment by means of component management. The efficiency and the changeability of software component based service system implementations can rise substantially, if the software component structure is a real refinement of the service structure supporting additional opportunities for component reuse. As a consequence, however, more rich dependency relations arise since each software component may depend on certain versions of other ones. Kon et.al. [17] propose the utilization of component configurators which maintain and manage lists of dependency hooks and client dependency references in order to cope with the relevant dependence problems and their implications for the reliability of complex distributed software systems. Chen [18] proposes procedures for the monitoring, analysis and reconfiguration of component structures to adequately address the dynamic reconfiguration of a complex system. Component replacement is the followed approach to tackle any reconfiguration issues.

It is worth mentioning that the messages exchanging in smart grids should be taken into consideration when studying them. Based on the literature, the dominant standards are the following three: i) Data Distribution Service framework (DDS) [19]; ii) Extensible Messaging and Presence Protocol (XMPP) [20]; and iii) RabbitMQ [21]. By carefully analyzing the aforementioned frameworks we reached the conclusion that the QoS capabilities of XMPP are limited which are mostly supported by protocol extensions. On the other hand,

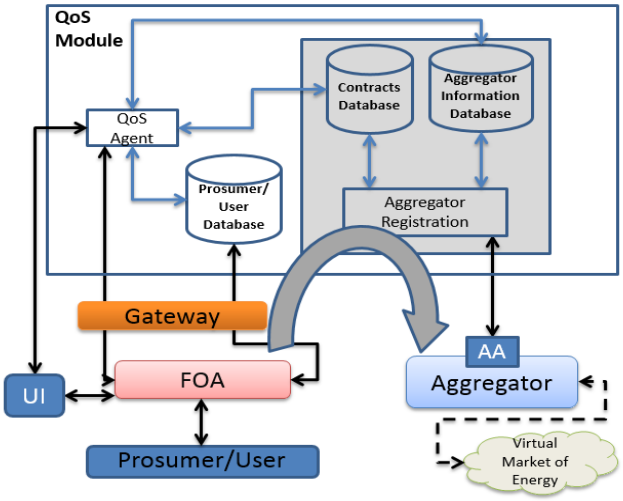

Fig. 2. Proposed QoS Architecture

DDS targets distributed real-time systems and therefore it is capable of addressing very complex distributed applications, where QoS requirements have to be guaranteed, while,

RabbitMQ is used for high performance distributed system applications, and is mostly focused on high performance (not predictability). It is therefore straightforward to conclude that DDS is the most suitable candidate for smart grid applications which come with high QoS requirements.

The challenges associated with the forecasting and demand response associated with energy usage were also discussed in [23]. Energy usage forecasting can be categorized into short-term, medium-term, and long-term forecasting. Hong et al. [24] adopted a multiple linear regression mechanism for conducting short-term forecasting, which provides an interpretability of the behavior of the electricity usage in the service territory. A semi-parametric additive model proposed by Fan et al. in [25] used a regression mechanism and investigated the nonlinear relationships between energy usage data and variables in the short-term time period. In addition, a human-machine construct intelligence framework was proposed in [26] to determine the horizon year load for a long term load forecasting. Machine learning methods such as SVMs and neural networks have been used in carrying out forecasting [27, 28, 29, 30, 31, 32, 33, and 34]. For example, Shi et al. [28] developed a SVM-based model for one-day-ahead power output forecasting using the characteristics of weather classification. Research has been conducted in predicting energy consumption for smart homes. In [35], a method for predicting energy usage using data collected from CASAS Smart Environment System is introduced. People's activities, overall movement in the home, and frequency of sensor data events are used to predict energy usage.

III. Proposed QoS Architecture

The QoS architecture presented in the paper consists of the following components: the Aggregator [4], the Aggregator Agent (AA), the Prosumer/User [4], the Flex-Offer Agent (FOA) [4], the QoS Agent, the Aggregator Registration, and databases: to store information regarding the Prosumers/Users, the Contracts (closed, served, etc.), and information regarding the available Aggregators and their characteristics. See Fig. 2.

The Prosumers/Users send their micro flex-offers to the Aggregator, through the FOA and QoS Module. A micro flex-offer states the possibility of a Prosumer/User to consume a certain amount of energy and the time interval during which it has the flexibility to schedule that consumption. There is also the possibility the flex-offer to be generated by the Flex-Offer Agent or by a Flex-Offer Agent that resides on the Aggregator's side, but we will not consider these two options in the present work.

The Aggregators are capable of joining several micro flex-offers into larger macro flex-offers, which are then placed on the electricity market. The energy market will answer with bids to buy and sell energy at given times. Aggregators receive and respond to the bids which allocate energy consumption periods to the macro flex-offers. After, they disaggregate macro flex-offer responses and send an answer to the Prosumers/Users which specify the periods of time to consume the required energy amount from the grid at a lower cost. It is the QoS Module that has the responsibility to find the best matching between the Prosumer's request for a service and the Aggregator that best covers its needs, in terms of response time, availability, and cost.

A. *Aggregator*

The Aggregator is responsible for the handling of flex-offers from the FOA, joining (aggregating) several micro flex-offers into a larger macro flex-offer, placing the macro flex-offer on the Virtual Market of Energy, disaggregating scheduled macro flex-offers, sending scheduled micro flex-offers to FOAs, controlling the execution of a micro scheduled flex-offer, determine if the execution of the flex-offer by the Device had been done according to the scheduled flex-offer. Each Aggregator can be specialized on different types of devices, by running the most adequate algorithms for the aggregation and disaggregation of flex-offers.

B. *Aggregator Agent (AA)*

Every Aggregator has an agent that provides information to the QoS Module. The Aggregator provides information to the QoS Module that has to do with the number of the users it is able to serve, possible cost of the provided service, time to respond to the Prosumer's request. It can also provide information regarding the type of services it can provide. It is common in the energy market to have a range of different type of Aggregators to cover the needs for home appliances (e.g., washing machines, heat pumps, etc.) and different ones to cover the needs of Electric Vehicles charging. This information is of great importance to the QoS Module in order to correctly and fast identifies the most appropriate Aggregator to deliver the service to the Prosumer. The AA is indirectly connected to Flex-Offer Agent (FOA) via the QoS Module.

C. *Aggregator Registration*

The Aggregator Registration allows Aggregators, through

the Aggregator Agent (AA) to submit: their id, service descriptions, cost functions, availability, and number of Prosumers/Users they can serve, to the QoS Module.

D. Prosumer/User

A Prosumer (or User) owns devices and has an agreement with an Aggregator regarding utilizing the devices power consumption or production flexibility. Devices are the end equipment that consume or produce the energy belonging to a flex-offer, e.g., an EV, a heat pump or a washing machine. Devices can have the capability of being remotely controlled or might not have any computer interfacing capabilities. The Prosumer has to set up all relevant constraints/comfort requirements, which the flex-offer must fulfill. The Prosumer might be a household, factory, an office building, i.e. a legal entity that owns devices. A Prosumer uses a Flex-Offer Agent to generate flex-offers or it can configure these parameters through a user interface.

E. Flex-Offer Agent (FOA)

Every Prosumer/User has an agent that provides information to the QoS Module. FOA is a software module, which acts as an intermediate between Devices and Aggregators, being able to be executed on a variety of hard-ware platforms and easily configured to use different protocols. Based on constraints set up by the Prosumer and on power consumption measurements taken from devices it uses a specific algorithm to automatically generate micro flex-offers.

Other inputs like weather forecasts might also be used. The FOA can send the micro flex-offers to the Aggregator and receive the micro scheduled flex-offers from it. Another kind of information the Prosumer/User passes to the QoS is the type of service it needs (domestic appliances, heat pumps, or EVs). As in the case of the Aggregator Agent, this information is of great importance to the QoS Module in order to correctly and fast identifies the most appropriate Aggregator to deliver the service to the Prosumer. The Flex-Offer Agent passes the request for a service to the QoS Module through the QoS Agent.

F. QoS Agent (QA)

QoS Agent (QA) is responsible for evaluating the Prosumer request, and identifies an Aggregator that properly meets the client's needs. The QoS Agent receives the request from the Flex-Offer Agent (FOA) and evaluates the Prosumer/User request against each available Aggregator in order to identify the one that best fits the Prosumer/User needs. A Prosumer's request will probably contain a service type, cost constraint and the preferred comfort level. Once the time the mapping is succeeded the micro flex-offer is passed to the Aggregator to continue with the building of the macro flex-offers and the placement to the market of energy.

G. User Interface

The User Interface can take care of the interactions among the Prosumers/Users, the FOA, and QoS Agent through a web-based interface. It can be used to allow generation of flex-offers by a Prosumer/User or just to enforce attributes like a particular comfort level to the QoS Module.

H. Gateway

The Gateway can be seen as a device that converts between the protocols used internally on a Home Area Network and the internet. It is possible to have the capability of executing the Flex-Offer Agent.

I. Contracts and Aggregator Information Databases

The Contracts Information Database is a database to store SLAs, closed, scheduled, and served contracts. The Aggregator Information database is a database for keeping information regarding the Aggregators, Aggregator's information like type of services, availability, response time and cost models. Also the id of the Aggregator is stored on the Aggregators Information database. The id of the Aggregator is important in order the Prosumer/User through the FOA, and the QoS Module to identify the correct one.

J. Prosumer/User Database

The Prosumer/User Database is a database that holds information regarding the Prosumers/Users. Information like: power consumption, type of Prosumer/User (flex-offer enabled or legacy device), if he was served or not.

K. QoS Module Interactions

The available Aggregators register themselves to the QoS Module and particular to the Aggregators Information Database, providing information like type of provided services, response time and cost models. The Prosumer asks for a service, which in our case is a need for energy consumption. This type of information is named micro flex-offer. It is then the responsibility of the QoS Module to perform all the needed steps in order to spot the Aggregator that best serves the needs of the Prosumer. Figure 3 (see p. 5) presents the interactions between the Prosumer, the QoS Module and the Aggregator:

1. Aggregators register themselves (with their id), and their services (type of services, response time, cost models, and number of Prosumers/Users each can serve) with the QoS Module.

2. A Prosumer/User initiates the sequence of steps, by sending to the QoS Module a QoS request (pointing out the requested service type, amount of needed energy, cost constraints, time flexibility).

3. The QoS Module identifies the Aggregator that best fits the needs of the Prosumer/User. The QoS Module creates a token that includes information like the id of the Aggregator, a session id, the service id, expiration date and time for the offer.

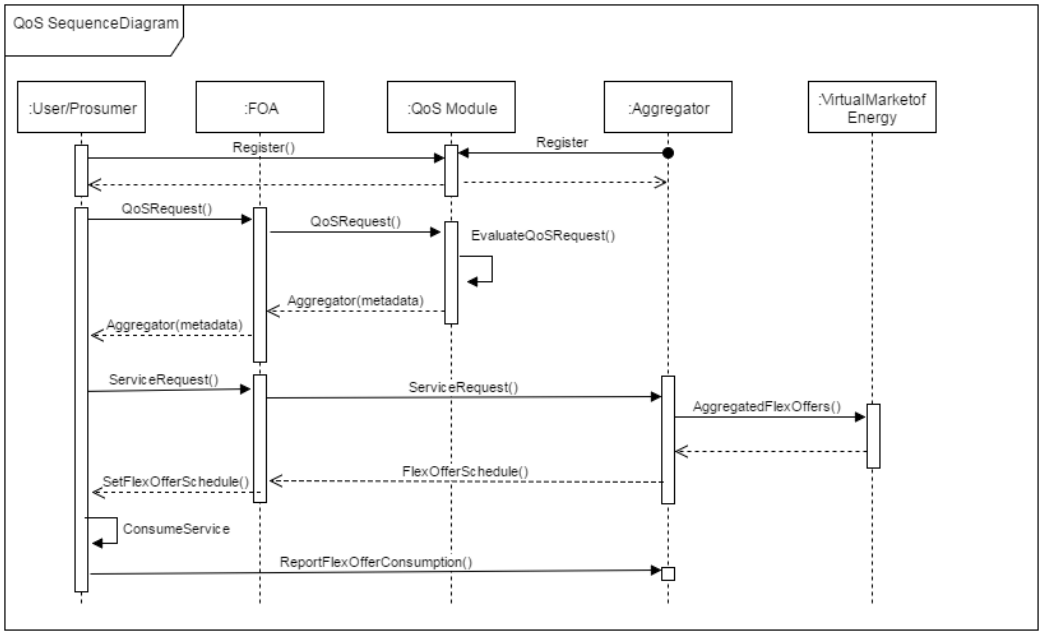

Fig. 3. Proposed QoS Architecture

4. If the Prosumer accepts the offer, the QoS Module saves it in the Contract database. The Prosumer only needs the created token to request the service in the given time.

5. The Prosumer makes a service request to the Aggregator using the created token.

6. The Aggregator creates the macro flex-offer and places a bid to the Virtual Market of Energy. The market answers back with a schedule.

7. The Aggregator sends the Schedule to the Prosumer/User, through the Flex-Offer Agent.

8. The Prosumer consumes the service and reports back to the Aggregator the power consumption.

IV. Applying Mining & Machine Learning to Smart Grid

In classical machine learning, the complexity and diversity of the field is controlled by the "Black-box" principle, where each machine learning method is expected to fit a simple mold. We will try to provide some insight to the "Black-box" in order to present its architecture and functionality. The Query Results Management (QRM) component/module will be responsible for managing the data that are extracted from the queries to the QoS system and for assembling the dataset that will be fed to the machine learning algorithm. In Figure 4, an illustration of where the QRM manager component is situated in relation to QoS's system and to the Machine Learning module is presented. The QRM module, a fundamental component of a more complete system, will be responsible for supporting the following functionalities:

1. Establishing a safe connection to the QoS databases;
2. Querying the databases, receiving the data; and
3. Saving the data in a file and in the proper format for the Machine Learning Management (MLM) module.

Even though many of the algorithms are different there are some common steps that should be followed while developing and applying a machine learning algorithm. These needed/common steps are the following:

1. **Data Collection:** Meaning the method for collecting the data. It varies from obtaining the data through an API, RSS feed, or even a device that collects data and sends them to you, etc.
2. **Data Preparation:** Making sure that the data are in a usable format. Some algorithms need features in a special format. Some can deal with features and variables as strings, and some others need them to be transformed into integers.
3. **Training the algorithm:** In this step, you feed the algorithm with "clean" data from the previous steps and obtain knowledge and insight from the data. In the case of unsupervised learning, there is no training step, since there is no target value.
4. **Testing the algorithm:** In this step, the evaluation of the algorithm takes place. In the case of supervised learning, you have known values for evaluating the algorithm (i.e. you have examples of data known as the ground truth that you can check against the performance of the algorithm). In unsupervised learning, there is a need to use other metrics like

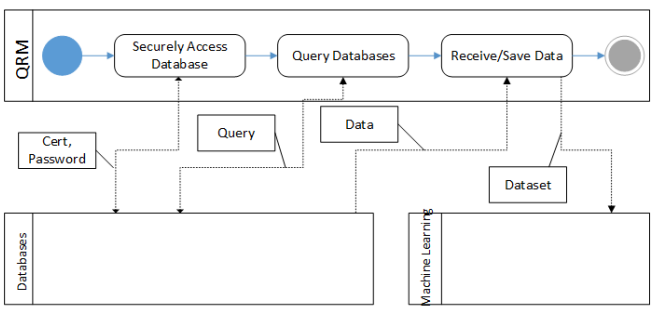

Fig. 4. QRM Component's Interactions

support and confidence to evaluate its success.

5. **Usage:** The actual implementation of the algorithm in practice that includes all the previous steps. There is also a need to continuously check if all the previous steps are working as expected. The QRM component will be responsible for the two first steps of the aforementioned procedure.

A. *Query Results Management (QRM) Module*

The QRM module will be responsible for the two first steps of the aforementioned procedure. Figure 5 presents the interactions of the QRM module with the Databases and the Machine Learning Module.

QRM will communicate with the RDF Database to query it and get the results. The results could be in XML, JSON, CSV or TSV format. The QRM Manager Component will be responsible to Securely Accessing the QoS's Databases by setting up a two way Secure Sockets Layer (SSL) connection to the Apache Jena Fuseki server [22] in order to securely query the RDF databases.

TABLE I

Setup two way SSL

```
System.setProperty("javax.net.ssl.keyStore",
"./keystores/fuseki.jks");
        System.setProperty("javax.net.ssl.tr
ustStore", "./keystores/truststore.jks");
        System.setProperty("javax.net.ssl.ke
yStorePassword", "********");
        System.setProperty("javax.net.ssl.tr
ustStorePassword", "password");
```

TABLE II

Execute a SPARQL query against the endpoint

```
final String serviceUri =
"https://test.uoe.com:9000/servicePaths/quer
y";
                final String query = "SELECT
?userUri ?typeOfService ?serviceAvaliability
?serviceCost ?aggregatorUri\n" +
"\n" +
"WHERE {\n" +
"\n" +
"?userUri
<http://www.uoe.com/ontology#completedBy>
?userUri .\n" +
"\n" +
"?typeOfService
<http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> ?typeOfService .\n" +
"\n" +
"?serviceAvaliability
<http://www.w3.org/2000/01/rdf-
schema#subClassOf>+
<http://www.uoe.com/ontology#serviceAvaliabi
lity> .\n" +
"\n" +
"?serviceCost
<http://www.uoe.com/ontology#associatedWith>
?typeOfService .\n" +
"\n" +
"?aggregatorUri
<http://www.uoe.com/ontology#basedOn>
?aggregatorUri .\n" +
"\n"
"}";
```

B. *Machine Learning Management (MLM) Module*

The MLM module will be responsible for the three last steps of the aforementioned procedure (steps 3 to 5). It will provide the needed functionality for the system to be in position to get the clean data, pass them to the machine learning algorithm and return useful conclusions. The MLM module should be in position to find interesting relationships in a large dataset. Quantifying interesting relationships is twofold. The first way is a frequent itemset, and the second is the one measuring interesting relationships in association rules.

One such approach is the Apriori [12] algorithm. Apriori uses the so-called Apriori principle to reduce the number of sets that are checked against the dataset. The Apriori principle denotes that if an item is infrequent, then supersets containing that specific item will be infrequent too. Apriori starts from single itemsets and creates larger sets by combining sets that meet the minimum support measure. Support is used to measure how often a set appears in the original dataset. Once frequent itemsets are found, someone may use them to generate association rules. The importance of an association rule is measured by the confidence. Confidence denotes the number that this rule applies to the frequent itemsets. The pseudocode of the Apriori algorithm is presented in Algorithm 1.

Algorithm1. The Apriori algorithm.

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

(1) L_1 = {frequent items};
(2) **for** (k = 1; L_k != ∅; k++) **do begin**
(3) C_{k+1} = candidates generated from L_k;
(4) **for each** transaction t in database do
(5) increment the count of all candidates in
(6) C_{k+1} that are contained in t
(7) L_{k+1} = candidates in C_{k+1} with min_support
(8) **end**
(9) **return** $\cup_k L_k$;

Another approach is the FP-growth [14] algorithm. The FP-growth algorithm is another efficient way of finding frequent patterns in a dataset. Even though it follows the Apriori principle, it is much faster than the Apriori one, since it goes over the dataset only twice. The data is stored in an FP-tree structure. Afterwards it is straightforward to find frequent

itemsets by finding conditional bases for an item, and eventually building a conditional FP-tree.
The aforementioned process is repeated, by conditioning on more items, until the conditional FP-tree has only one item. The pseudocode for the FP-growth algorithm is presented in Algorithm 2.

Algorithm2. The FP-growth algorithm.

Input: constructed FP-tree
Output: complete set of frequent patterns
Method: Call FP-growth (FP-tree, null).
Procedure FP-growth (Tree, α)
{
(1) **if** *Tree* contains a single path *P* then
(2) **for each** combination (denoted as β) of the nodes in the path *P* **do**
(3) generate pattern $\beta \cup \alpha$ with *support = minimum support of nodes* β
(4) **else for each** a_i in the header of *Tree* **do** {
(5) generate pattern $\beta = a_i \cup \alpha$ with *support* = a_i*.support;*
(6) construct β's conditional pattern base and then β's conditional FP-tree $Tree_\beta$
(7) **if** $Tree_\beta$!= ∅
(8) **then** call FP-growth ($Tree_\beta$, β)
}

C. Finding Frequent Itemsets

The support of an itemset is defined as the percentage of the dataset that contains this frequent itemset. Frequent itemsets are a collection of items that frequently occur together. For instance in our dataset, a series of interactions (users served, services asked, aggregators involved, etc.). In our specific case an itemset is having the following format:

{userId, serviceType, serviceAvaliability, aggregator Id, serviceProvider, serviceCost, responseTime}

Support applies to an itemset, so we can define a minimum support and get only the itemsets that meet that minimum support. Support can range from 0 to 1. The confidence is defined for an association rule like {User 1} → {Service 1}. The confidence for this rule is defined as support ({User 1, Service 1})/support ({User 1}). The support and confidence are ways someone can quantify the success of our association analysis. Let us assume we want to find all sets of items with a support greater than 0.6. We could generate a list of every combination of items and then count how frequently these occur.

D. Mining Association Rules from the Extracted Itemsets

To find association rules, we first start with a frequent itemset. Association rules suggest that a strong relationship exists between two items. We know this set of items is unique, but we want to see if there is anything else we can get out of these items. One item or one set of items can imply another item. From the dataset we have, if we have a frequent itemset, {User 1, Service 1, Cost 1}; one example of an association rule is Service 1 → Cost 1. This means if someone chooses Service 1 Cost 1, then there's a statistically significant chance that the User will choose Service 1. The converse does not always hold.

In Section IV.*C*, an itemset is quantified as frequent if it met our minimum support level. There is a similar measurement for association rules. This measurement is called the confidence. The confidence for a rule P → H is defined as support (P | H)/ support (P).

Similarly to the frequent itemsets generation in Section 4.3, we can generate many association rules for each frequent itemset. It would be desirable if we could reduce the number of rules to keep the problem tractable. We can observe that if a rule does not meet the minimum confidence requirement, then subsets of that rule also will not meet the minimum. We can use this property of association rules to reduce the number of rules we need to test.

V. Results

A series of tests performed in order to check the validity of the proposed approach that was first presented in [36]. Python used for the implementation of the machine learning proposed approach/architecture. The outcome of the association and data analysis that took place has shown that the proposed framework is in position to provide an insight on the behavior of the users - Prosumers, meaning how they interacted with the system and spot common patterns that lead or not to a successful completion of an asked service.

The results in this section contain the full information of the users that served by the available providers. The format is: {userId, serviceType, serviceAvaliability, aggregatorId, serviceProvider, serviceCost, responseTime}. From the above returned dataset and by applying data analysis someone can easily extract useful groups of characteristics per User, per Service, per Aggregator, etc., and combinations of them.

An example on how a visualization of the aforementioned results may be used can be found in Figure 6. The bar chart gives an insight on how well each requested service (blue part – providing the percentage) was served and by how many providers (orange part – providing the absolute number). For example the first from below requested service was served in a 100% and it was only provided by one service provider. This can be used by the Market of Energy for an in advance prediction of the customers' behavior to better arrange its production, thus avoiding any possible energy shortage and/or fluctuation.

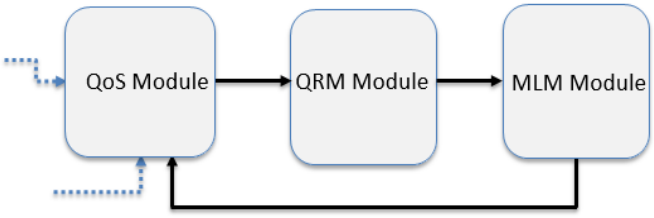

Fig. 5. Proposed QoS and Machine Learning Abstract Architecture

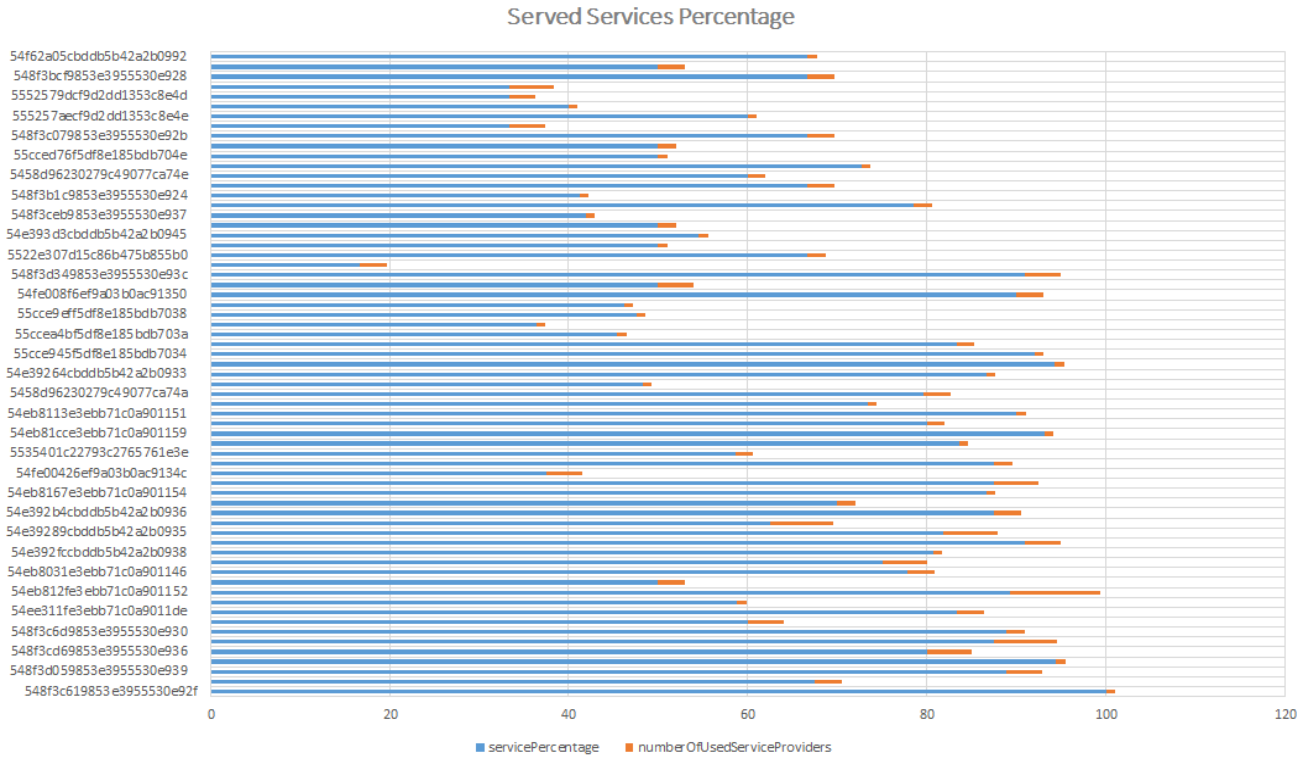

Fig. 6. Visualization of Served Service percentage Per Service Provider

VI. Conclusion

In this paper we presented a complete Quality of Service architecture targeting the Smart Grid world. All the involving parts were in detail described and documented. QoS attributes like: type of service to be served, response time, availability, and cost where taken into consideration while sketching the proposed architecture. Future work will include definition of algorithms to be used for the QoS provisioning and implementation of the proposed architecture. Another equally important step is handling the different ways that a flex-offer can be generated and come up with an as common as possible approach. In this paper we considered the flex-offer to be created by the Flex-Offer Agent that is actually connected to the Prosumer/User. Other identified formal cases are the generation of the flex-offer on the Aggregator, by using power measurement data available on the cloud, and the flex-offer to be initiated by the Prosumer/User, through a User Interface provided by the Flex-Offer Agent.

We also presented an initial supplementary architecture to mine the information stored in the databases and further process the data with the use of machine learning algorithms to extract useful information, like identifying common patterns amongst multiple users/prosumers. Some initial results from the proposed approach were also presented. An unsupervised approach based on the Apriori algorithm was used to serve our needs. Common patterns for instance in electricity usage in terms of time and amount. In this way the market of energy will be in position to better regulate its production thus leading to a more stable and economically sustainable power grid. Possible supervised approaches based on neural networks, random forests or Support Vector Machines (SVMs) should also consider in order building a multilevel and an autonomous as possible predictive model.

Acknowledgment

The authors wish to thank the former WP5 partners from the Arrowhead project [13], for sharing their views and thoughts on defining a QoS architecture for the Smart Grid. Part of the work was also supported by the HEFCE UK Catalyst project.

References

[1] E. Newcomer, G. Lomow, “Understanding SOA with Web Services,” ISBN-10: 0321180860, ISBN-13: 9780321180865, Publisher: Addison-Wesley Professional, Copyright: 2005.

[2] M. Rosen, B. Lublinsky, T.K. Smith, J.M. Balcer, Applied SOA: Service-Oriented Artchitecture and Design Strategies. John Wiley & Sons; Pub. Date: June 16, 2008 , Print ISBN: 978-0-470-22365-9; Web ISBN: 0-470223-65-0, 2008.

[3] E. Thomas, SOA Design Patterns. Prentice Hall PTR, ISBN: 0136135161, 2009.

[4] L.L. Ferreira, L. Siksnys, P. Pedersen, P. Stluka, C. Chrysoulas, T. Guilly, M. Albano, A. Skou, C. Teixeira, T. Pedersen, “Arrowhead compliant virtual market of energy,” in Emerging Technology and Factory Automation (ETFA), 2014 IEEE, Sept 2014, pp. 1–8, 2014.

[5] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy, “A distributed resource management architecture that supports advance reservations and coallocation,” in: Proc. Intl. Workshop Quality of Service 1999, UCL, 1–4 June, London, 1999, pp. 27–36, 1999.

[6] S. Ran, “A model for web services discovery with QoS,” ACM SIGEcom Exchanges 4 (1) 1–10, 2003.

[7] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, J. Schiller, "A concept for QoS integration in web services," in: Fourth Intl. Conf.Web Information Systems Engineering Workshops, WISEW'03, Roma, Italy, December 2003, pp. 149–155, 2003.
[8] F. Siqueira, V. Cahill, "Quartz: A QoS architecture for open systems," in: The 20th Intl. Conf. Distributed Computing Systems, ICDCS 2000, 10–13 April, Taipei, Taiwan, 2000, pp. 197–204, 2000.
[9] D.L. Tien, O. Villin, C. Bac, "Resource managers for QoS in CORBA," in: Second IEEE International Symp. Object-Oriented Real-Time Distributed Computing, 2–5 May, Saint-Malo, France, 1999, pp. 213–222, 1999.
[10] T. Cucinotta, A. Mancina, G. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, F. Rusinà, "A real-time service-oriented architecture for industrial automation," IEEE Trans. Ind. Informat., vol. 5, no. 3, pp. 267-277, 2009.
[11] C. Cavanaugh, L.R. Welch, B. Shirazi, E. Huh, S. Anwar, "Quality of service negotiation for distributed, dynamic real-time systems," in: IPDPS Workshop on Bio-Inspired Solutions to Parallel Processing Problems, BioSP3, 15 April, Fort Lauderdale, FL, 2002, pp. 757–765, 2002.
[12] R. Agrawal, R. Srikant, "Fast algorithms for mining association rules," in: Proc. 20th Int. Conf. Very Large Data Bases, VLDB, 1994, pp. 487–499, 1994.
[13] The Arrowhead project: http://www.arrowhead.eu/ [accessed December 2016]
[14] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Chang, "PFP: Parallel FP-Growth for Query Recommendation," RecSys 2008, Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 107-114, 2008.
[15] M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, "Service oriented computing: State of the art and research challenges", Computer, no. 11, pp. 38–45, 2007.
[16] C. Chrysoulas, G. Kostopoulos, E. Haleplidis, R. Haas, S. Denazis, O. Koyfopavlou, "A decision making framework for dynamic service deployment", in Proc. 15th IST Mobile & Wireless Communications Summit, Mykonos, Greece, volume 3, 2006.
[17] F. Kon, R. H. Campbell, "Dependence management in component-based distributed systems", IEEE concurrency, no. 1, pp. 26–36, 2000.
[18] X. Chen, "Dependence management for dynamic reconfiguration of component-based distributed systems", in 17th IEEE International Conference on Automated Software Engineering. Proceedings. ASE 2002, pp. 279–284, 2002.
[19] Object Management Group, Inc. (OMG): Data Distribution Service for Real-Time Systems Specification, Version 1.1, formal/05-12-04, December 2005.
[20] P. Saint-Andre, K. Smith, R. Tronc, "XMPP: The Definitive Guide", O'Reilly, 2009.
[21] J. Videla, J.W. Williams, "RabbitMQ in Action: Distributed Messaging for Everyone", MEAP Edition, Manning Early Access Program, 2011.
[22] Fuseki: serving RDF data over HTTP. https://jena.apache.org/documentation/serving_data/, 2016.
[23] P. Luh, L. Michel, P. Frieland, C. Guan, and Y. Wang. Load forecasting and demand response. In Proceedings of IEEE Power and Energy Society General Meeting, pages 1–3, 2010.
[24] T. Hong, M. Gui, M. Baran, and H. Willis. Modeling and forecasting hourly electric load by multiple linear regression with interactions. In Proceedings of IEEE Power and Energy Society General Meeting, pages 1–8, 2010.
[25] S. Fan and R. J. Hyndman. Short-term load forecasting based on a semi-parametric additive model. IEEE Transactions on Power Systems, 27(1):134–141, 2012.
[26] T. Hong, S. Hisiang, and L. Xu. Human-machine co-construct intelligence on horizon year load in long term spatial load forecasting. In Proceedings of IEEE Power and Energy Society General Meeting, pages 1–6, 2009.
[27] Z. A. Bashir and M. E. El-Hawary. Applying wavelets to short-term load forecasting using pso-based neural networks. IEEE Transactions on Power Systems, 24(1):20–27, 2009.
[28] J. Shi, W.-J. Lee, Y. Liu, and Y. Yang. Forecasting power output of photovoltaic systems based on weather classification and support vector machines. IEEE Transactions on Industry Applications, 48(3):1064–1069, 2012.
[29] Z. Yun, Z. Quan, and S. Caixin. Rbf neural network and anfis-based short-term load forecasting approach in real-time price environment. IEEE Transactions on Power Systems, 23(3):853–858, 2008.
[30] Y. Wang, Q. Xia, and C. Kang. Secondary forecasting based on deviation analysis for short-term load forecasting. IEEE Transcation On Power Systems, 26(2):500–507, 2011.
[31] M. Afshin, A. Sadeghian, and K. Raahemifar. On efficient tuning of ls-svm hyper-parameters in short-term load forecasting: A comparative study. In Proceedings of IEEE Power Engineering Society General Meeting, pages 1–6, 2007.
[32] W. Li and P. Choudhury. Probabilistic planning of transmission systems: Why, how and an actual example. In Proceedings of IEEE Power and Energy Society General Meeting Conversion and Delivery of Electrical Energy in the 21st Century, pages 1–8, 2008.
[33] T. Hong, P. Wang, A. Pahwa, M. Gui, and S. M. Hsiang. Cost of temperature history data uncertainties in short term electric load forecasting. In Proceedings of International Conference on Probabilistic Methods Applied to Power Systems, pages 212–217, 2010.
[34] P. Pinson, C. Chevallier, and G. N. Kariniotakis. Trading wind generation from short-term probabilistic forecasts of wind power. IEEE Transactions on Power Systems, 22(3):1148–1156, 2007.
[35] C. Chen, B. Das, and D. J. Cook, "Energy prediction based on resident's activity," in Proceedings of the International workshop on Knowledge Discovery from Sensor Data, 2010.
[36] C. Chrysoulas, and M. Fasli, "A service oriented QoS architecture targeting the smart grid & machine learning aspects", in SpliTech 2016, IEEE Co-Sponsored International Multidisciplinary Conference on Computer and Energy Science, 13-15 July, Split, Croatia, 2016.

Christos Chrysoulas received his Diploma and his Phd in Electrical and Computer Engineering from the University of Patras in 2003 and 2009 respectively. During his Phd (2004-2009) and PostDoc studies (2010-2015) his research was focused on Machine Learning, Big Data, E-Learning systems, Computer Networks, High Performance Communication Subsystems Architecture and Implementation, Wireless Networks, New Generation Networks Architectures, Service Oriented Architectures (SOA), Resource Management and Dynamic Service Deployment in New Generation Networks and Communication Networks, Grid Architectures, Semantics, Semantic Grid, Smart Grids, and IoT. He joined CISTER Research Center as an Invited Researched in 2013. He joined University of Porto as Post-Doc Research fellow in 2014 and from July 2015 he is with the University of Essex, holding a Senior Officer Researcher position. The outcome of this effort was properly announced in more than 25 technical papers (1 Book Chapter, 6 Journal Papers, and 19 Conference papers) in these areas.

Dr. Chrysoulas participated as Senior Research/Engineer in both European and National Research Projects, and also participated in the OPC Standardization Foundation.

Maria Fasli received her BSc in Informatics from the Technological Education Institution, Department of Informatics, Thessaloniki, Greece in 1995 and her PhD in Computer Science from the University of Essex in 2000. She joined the Department of Computer Science at the University of Essex as a Senior Research Officer in August 1999 and took on the position of Lecturer in 2000. She became Senior Lecturer in 2007 and Professor in 2012. She served as Head of School of Computer Science and Electronic Engineering between 2009-2014. Since 2014 she has been the Director of the Institute for Analytics and Data Science at the University of Essex. Her research interests lie in artificial intelligence, agents and multi-agent, machine learning, data exploration, analysing and modelling complex data (structured and unstructured), Big Data, as well as semantics-based techniques for user profiling and adaptation including modelling context. She has published over 110 papers in the field of artificial intelligence and multi-agent systems and has been involved in organising/chairing international events.

Prof. Fasli was awarded a National Teaching Fellowship (NTF) by the Higher Education Academy (HEA) UK for her innovations and contributions to education and supporting the student experience in 2005, and in 2016, she was awarded the first UNESCO Chair in Analytics and Data Science.