Data Quality-Based Resource Management in Enterprise Service Bus

by

Kamyar Rasta

Supervisors:

Professor Dr. Andreas Prinz
Trinh Hoang Nguyen

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree Master of Science in Information and Communication Technology.

University of Agder
Faculty of Engineering and Science
Department of Information and Communication Technology

Grimstad, Norway
June 2013

Abstract

Enterprise Service Bus (ESB) is proposed to address the application integration problem by facilitating communication among different systems in a loosely coupled, standard-based, and protocol independent manner. Data sources are maintained out of the ESB's control and there should be a mechanism to select the most suitable data source among all available data sources, especially, when two or more data sources are about the same object. For instance, it is normal to use more than one sensor to measure pressure or temperature at a particular point. Data quality can play an important role in selecting data sources in ESB because the quality of data is an essential factor in the success of organizations. There is no built-in component in the current ESB platforms to handle data quality. In this thesis, we present a flexible and comprehensive data quality framework for managing resources in ESB. The framework is independent of ESB platforms. We evaluate our framework using five different scenarios within the wind energy domain.

Keywords: Data quality, information quality, enterprise service bus, resource management.

Dedicated to my parents, Mousa and Homa

Preface and Acknowledgements

This Thesis was submitted in partial fulfillment of the requirements for the degree Master of Science in Information and Communication Technology (ICT) at the University of Agder where the workload is set to a total of 30 ECTS credits. The work has been done in the period from January to June 2013.

I would like to express my sincere gratitude to my supervisor Prof. Andreas Prinz. His careful supervision, encouragement, inspiration, visionary ideas, and constant support helped me a lot in moulding my efforts into favorable outcomes.

I am very grateful to my supervisor, Trinh Hoang Nguyen, who is a Doctoral Fellow of the ICT department in University of Agder. He has dedicated countless hours with me in discussion, attending practice talk, and in general, ensuring an excellent environment.

Last, but certainly not the least, I would like to express my gratefulness and humility to my parents and my sister for their support during the Master thesis.

Kamyar Rasta
June 2013
Grimstad, Norway

Contents

List of Figures

List of Tables

Abbreviations

AJAX	Asynchronous Javascript and XML
DSMS	Data Stream Management System
ESB	Enterprise Service Bus
EDA	Event Driven Architecture
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
RDBMS	Relational Database Management System
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TDQM	Total Data Quality Management
UML	Unified Modeling Language
WSDL	Web Services Description Language
XML	Extensible Markup Language

Terminology

Data quality dimension: an attribute of data. It shows the criterion, view, and attribute measurements of the data.

Data quality: the extent to which attributes of data are suitable for their use. The data quality is a multidimensional notion.

Dynamic service invocation: the act of putting an instance of a service into action at run-time when the service is invoked by service consumer.

Integration Server: a logical server consists of a couple of servers. They cooperate to perform the task of data integration.

Quality Manager: a high level component deals with selecting the data source according to the quality criteria.

Quality-based selection: The process of selecting a data source out of existing sources with respect to the quality dimensions.

Service registry: it refers to a repository where stores the information of service descriptions.

Service: a multi-step action performed by a server. It can has several operations with parameters and values.

Static service invocation: the act of putting an instance of a service into action using pre-generated stubs.

User request: a computational task to be done such as getting the temperature in a specific time frame.

Chapter 1

Introduction

1.1 Background and Motivation

The growing number of applications distributed across the Internet, interacting with a large number of users has led to a need for a better communication platform between users and applications. Service-Oriented Architecture (SOA) increasingly gains momentum in industry as a means to facilitate communication and collaboration between different systems running on multiple platforms. SOA is based on loosely coupled, distributed, and interoperable services [1]. Services are collections of well-structured business functionalities that are exploited to build a software system. An enterprise application that is built based on SOA concepts contains a collection of interacting users and services [1, 2].

Figure 1.1 shows the main components of the traditional point to point SOA. A service provider publishes service descriptions. Service descriptions are registered in a service registry. Later a service consumer finds a previously published description of a service in the service registry. When the description has been retrieved, the service consumer invokes the service from the service provider [1, 3].

All components in Figure 1.1 interact with each other through a direct connection. However, a drawback of this approach is that once the number of providers and consumers increases, the whole system is faced with a significant complexity. Any changes in one component might be disseminated through the whole system because new connections between that component and others need to be reestablished. The lack of flexibility increases the cost of the development and maintenance especially when changes occur frequently.

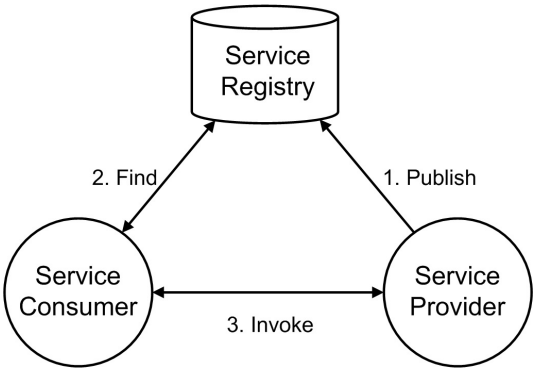

FIGURE 1.1: SOA' components traditional point-to-point communication approach (simplified) [1, 3].

SOA provides a request/reply pattern where the service consumer has to wait for the service provider to complete the requested operation [4]. However, many integration problems follow a different pattern where the service consumer shows its interest in a specific event and subsequently is notified when the service provider generates events [5]. The Event-Driven Architecture (EDA) has been introduced to handle such a pattern. EDA is a software architecture for designing and developing applications in which events transmit among system components and services [6]. An event is a significant change which occurs in the system state. When an event occurs, it is propagated to all interested parties who have subscribed to that event.

Combination SOA and EDA paradigms provides a foundation for an emerging technology that offers a unified backbone on top of which enterprise services can be deployed, planned, and executed [3, 7]. This backbone is called Enterprise Service Bus (ESB) that is a messaging infrastructure for integration which enables the implementation, deployment, and management of applications following SOA paradigms [8].

ESB supports a large number of communication patterns over different transport protocols [8]. ESB has two key objectives. The first one is that ESB makes a loosely coupled integration solution which refers to a system in which components make a little use of other components. Consequently, components are isolated from others and the whole system becomes more reusable and adaptable to changes. The second objective is to divide the integration logic into separate and easy to manage parts [9, 10].

Figure 1.2 shows how ESB provides a centralized communication model for integrating service providers and consumers by avoiding direct communication between them. In this model each application first connects to a bus, and then all connected applications can

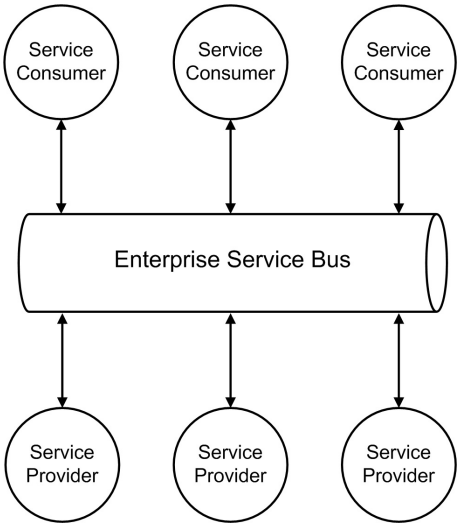

FIGURE 1.2: Enterprise Service Bus components communication approach (simplified) [1, 3].

share data by producing or consuming data on the bus [10]. The most relevant benefits of using ESB are listed as follows:

- *Standardized connectivity*: A large number of standard technologies are supported by ESB platforms for enterprise application integration.
- *Scalability*: ESB offers a loosely coupled architecture which provides scalability benefits such as high reliability, fault tolerance and transparency.
- *Routing*: As a message broker, ESB presents different levels of message routing and mediation [11, 12].

ESB can be employed as a centralized solution to solve the data integration problem effectively in various domains such as power system, eHealth, and oil & gas. In our previously published works [13–16], we use ESB for data integration in wind energy and eHealth domains. Those ESB-based data integration works are our starting point in this thesis. Although ESB handles communication between applications, it does not support a way to select the most suitable data source among all possible options.

As data become the main resource of organizations, many of them believe that quality of their data is essential to their success [17]. However, in small number of them this belief leads to action. In most organizations poor quality of data affects their business significantly [18] and they have influenced by the wrong decisions made according to the

data with the insufficient quality [19, 20]. Therefore, more and more organizations try to overcome their data quality problems.

Although there is no consensus on the definition of data quality [21], the data quality concept is defined as the extent to which the data is fit to reality and it is usually evaluated from consumer point of view [19]. Organizations assess, monitor, and improve the data quality to satisfy their consumers with proper data. In short, data quality handling has become an important factor in organization's success.

In the following section, we describe the research challenges addressed in this work and propose a coherent framework to demonstrate and compose these issues and their contributions.

1.2 Problem Statement

The starting point of this thesis is the need of a new mechanism for resource selection in ESB. One selection method is the use of data quality. Data quality-based resource management in ESB incorporates research from different areas such as data quality, service selection, and data integration based on ESB. This section gives an overview of these areas and a list of challenges that should be overcome in each of area to enable data quality-based resource management in ESB. The thesis specifically addresses the following problems:

- *Data source selection according to the quality criteria*: A widespread issue in data integration is the management of data with insufficient quality. For example in offshore wind energy, a couple of sensors are deployed on a windmill and they frequently measure and deliver the data to the users and applications by means of services. As sensors are prone to failures their results might be inaccurate, incomplete, and inconsistent [22]. Therefore, the data quality issues should be handled in such a way that users and applications can specify the desired quality level of the data. Only when the data source has the requested quality descriptions it would be used for further processing.

  The user has an information model that gives him lots of available data with different data quality attributes. The user can specify his quality requirements and the system processes the request and gives him the proper data. In enterprise applications with a potentially large number of data sources with quality information, handling the data quality at user level is a challenging problem. To tackle this problem, the data

quality process should be moved from the user level to the middleware level (i.e. ESB). However, there is no data quality component in the current ESB platforms. Therefore, knowledge about specifications of such a component is desirable to ensure the quality of data sources in ESB.

- *Data sources combination*: Another issue is that sometimes none of the available data sources has the required quality information. In this case, by combining existing data sources it is possible to improve the quality of data to meet the user's requirement. The combination of data source is defined as the process of constructing a data source from existing data sources. The combination process inherently needs the combination of data, combination of quality information, and the combination algorithm which will be discussed in the next chapters. The main objective of data source combinations is to improve a data source with higher data quality according to the existing data sources.

1.3 Thesis Definition

This thesis proposes a framework for data quality handling, to tackle the need of a new data source selection method in ESB. This project specifically entails the following tasks:

1. Find a way to describe, measure, and compare the quality of data sources based on the existing data quality frameworks.
2. Design a a flexible and comprehensive data quality framework for managing data sources in ESB.
3. Propose an approach for selecting a data source with the most proper data quality according to given quality criteria.
4. Provide dynamic binding and invocation of resources in ESB in order to make the implementation details transparent to users and applications.
5. Define a mechanism for improving data quality by combining different data sources.
6. Develop a prototype proving the concepts within the wind energy domain.

1.4 Thesis Outline

The thesis consists of six chapters that are organized as follows:

Chapter 1 - Introduction
The first chapter identifies the need for a new feature in the development of applications that use ESB as the communication platform. It briefly introduces the idea of data quality-based resource management, in which data sources are selected automatically and assigned dynamically to requesters. This chapter concludes by outlining the contents of this thesis.

Chapter 2 - Core Concepts and Literature Review
This chapter starts with an introduction about the main concepts of the thesis such as ESB, data quality, and data sources and continues with analysing the related work in two primary areas: data quality dimensions, and resource selection in ESB. For each area several research works and their relation to our solution are presented. This chapter is intended to supply background information used in the next chapters.

Chapter 3 - Towards A Data Quality-based Framework
Chapter 3 describes the design of the system and highlights the necessity of the new design model for resource management in ESB according to the quality criteria by identifying limitations of existing approaches. It also proposes the framework architecture in terms of its main components and the operations provided by them.

Chapter 4 - Implementation
This chapter presents the implementation of the proposed architecture by providing the details of the system components and how they perform the desired functionalities.

Chapter 5 - Evolution and Results
Chapter 5 describes the results and experiments undertaken, which shows that the proposed framework for data quality handling is a viable approach. Furthermore, it explains five usage scenarios within offshore wind management.Finally, it is discussed whether the achieved results met the expected requirements or not.

Chapter 6 - Conclusions and Future Work

The last chapter highlights the primary contributions of this thesis and the conclusions reached. It points out areas with a potential for future research, in the context of ESB and data quality and resource selection.

1.5 Chapter Summary

This chapter started with background about main concepts in SOA and application integration based on ESB. The chapter is continued by highlighting the problem of data quality in organizations. In the problem statement, the two main problems of this thesis are shortly described: data source selection according to the quality criteria and data sources combination. The goals of this work are presented in thesis definition and the chapter is finished by summarizing the thesis outline.

Chapter 2

Core Concepts and Literature Review

The first chapter introduces some research problems concerning data quality-based resource management in ESB. In turn, the purpose of this chapter is to review related work in two main research categories that are relevant to the research problem: data quality frameworks and applications, and resource selection in SOA-based systems. Afterwards, the chapter elaborates the core terms of this thesis and provides a basis for the next chapters. It introduces the structure of data sources and their elements. Finally, different data quality dimensions are presented.

2.1 Literature Review

In this section, we review existing approaches according to the primary areas aligned with the contributions of this thesis: data quality frameworks and applications and the resource selection in SOA-based systems.

2.1.1 Data Quality Frameworks and Applications

At the beginning, it is important to have a clear definition of what the data quality is to have better understanding of these research problems and possible solutions. Traditionally, sometimes the data quality is defined only from the accuracy point of view, although many research shows that it should be defined beyond accuracy [19].

The data and information refer to different concepts that are used frequently. The data is a collection of raw and unorganized symbols that represents real world states. The information is the processed, organized, and structured data according to the given context [23, 24]. In literature, the data quality and information quality are often used interchangeably even though the data and information are different concepts. In this thesis data quality and information quality are synonym terms.

As we discussed in Chapter 1, there is no consensus on the definition of data quality [21]. One widely used definition for data quality is the data that is fitted to use by the data consumer [19]. The data quality is a multi-dimensional concept and the commonly identified dimensions are accuracy, completeness, timeliness, and consistency [18, 25, 26]. The data quality dimensions are explained in the next section in detail.

The data quality research is a branch of information system[1] research. Thus, the origin of the most important data quality research can be traced back into information systems research [27]. Since the awareness about the data quality have increased, researchers have focused on data quality frameworks [20, 26, 28, 29], data quality dimensions [18, 30, 31], data quality assessment [18, 30, 32], data quality management [33, 34].

We choose 10 research works for our review. These papers are illustrated along a timeline in Figure 2.1. The number of citation of each paper and how they are cited from other papers are shown in this figure.

We can see in Figure 2.1 that the paper DeLone and McLean [33] is the most cited one because the paper has reviewed several data quality studies and gave an overview of the research directions in the field.

Another significant work in data quality field is proposed by Wang and Strong [30] which is cited by most of the other subsequent papers. Authors in [30] propose a hierarchical framework to capture the data quality aspects in a consumer centric approach. In order to develop the framework, they conduct a two-stage survey and a two-phase sorting study. Using their framework, high quality of data can be achieved. According to the Figure 2.1, the extensive data quality research works were presented during the period from 1995 to 2005.

Authors in [18] extend the idea that was proposed [30]. They classify the data quality dimensions into four patterns: intrinsic, accessibility, contextual, representational data quality patterns. Using these patterns, they provide a foundation to understand how

[1] Information system refers to creating, collecting, filtering, and processing data by a person or an organizational process [18].

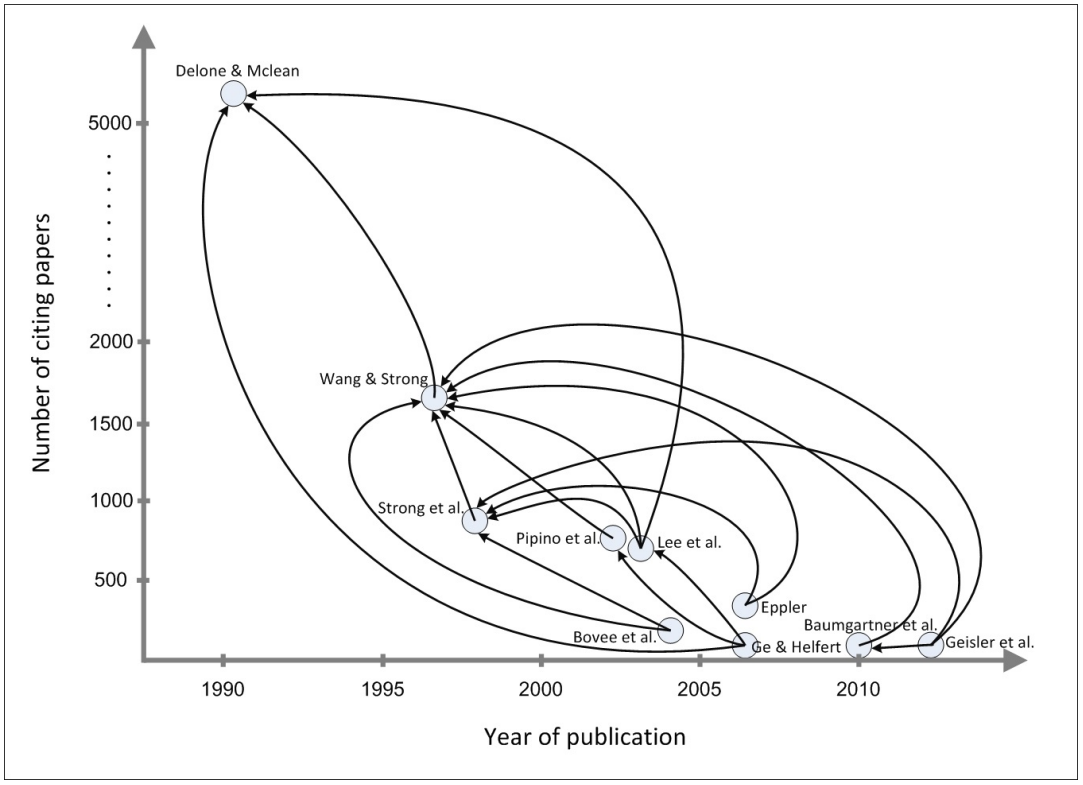

FIGURE 2.1: Overview of citations relationship among literature.

data quality problems should be developed in organizations. Consequently, they propose a way to improve the data quality in the organizations.

A disciplined approach to develop usable data quality dimensions are introduced by [31]. This work is mainly focused on answering to the question of how assess a companies' data quality. They describe subjective and objective assessments of data quality. They employ three function forms to develop data quality dimensions practically.

One of the primary works in data quality assessment has been carried out by [32]. They develop comprehensive methodology to assess and improve data quality. The methodology is called AIM quality (AIMQ) and consists of a model of information quality, a questionnaire, and analysis techniques. The questionnaire is used to measure the information quality. The analysis techniques are employed to interpret the information quality measures.

Based on a user-centric approach [30, 35], the authors in [28] propose a conceptual framework to assess overall information quality. Their information quality model has four

attributes: accessibility, interpretability, relevance, and integrity. This model provides a way of assessing the quality dimension. They also explore two algorithms to combine assessments into overall information quality measure.

As we mentioned, the data quality can influence the decisions made by the organizations. A framework is presented in [20] that uses data quality criteria to assess and analyze the decision quality. They specify and measure the relationship between accuracy and decision quality.

The situation awareness is defined as the perception of the environment in terms of time, space and other parameters to be able to predict the system state in the near future [36]. The authors in [29], improve situation awareness in traffic management systems by applying a methodology for assessing and improving the data quality.

Data Stream Management System (DSMS) is an application to handle continuous, rapid streams of data in real-time [37]. In [26] a framework is presented for data stream applications. The framework is based on ontology data model. The data quality process is performed by the metadata handled in ontology. The semantic definition of the data quality features of the DSMS is provided by ontology.

2.1.2 Resource Selection in SOA-based systems

There are some existing works related to resource selection in ESB that are interesting to consider.

Authors in [38] have proposed a multi-layered framework to support the content-based for intelligent routing path construction and message routing. Their approach facilitates the data source selection based on the message content. Dynamic service selection and composition in ESB have presented in [39]. The authors use abstract service names to define an Abstract Routing Path (ARP) and replace this path with uniform resource identifier (URI) of the real data provider at run-time. Their model makes the path selection in ESB transparent and service composition configuration dynamic. Authors in [40] have introduced a middleware representing a combination of SOA and Semantic Web and namely Semantic Service Bus. It enhances the processes of service discovery, routing, composition, etc., by employing semantic description of services.

The common fact in all aforementioned works is that the resource selection process is based on data itself. However, in our approach the resource selection is based on quality description of the data rather than the data itself.

2.2 Core Concepts

In this section we define the basic terms and concepts in the thesis and this part provides a basis for the next chapters. It defines the data sources and data quality dimensions which are the main components of this work.

2.2.1 Data sources

In our quality-based system, the users and applications look for a specific data to use. The data in this system is represented by an entity that is called a data source and this is one of the most important components of our system. A data source consists of three parts: data item, data item description, and quality description. These are described as follows:

- *Data item:* this is a sensor data. In particular, it is a stream of signals coming from a source. We call each signal in the stream a data point, for instance, 6 for wind speed at a specific date and time. The whole stream is called a data item, for example, 4000 wind speed values from the 1^{st} of May to the 1^{st} of June. The data item is provided by a service deployed on a data provider. The services can expose available sensor data to the users in a standard manner.

- *Data source description:* is the information about the data source to describe the data source in the context of the domain. The data item is only a sequence of numbers and there is no information of about what kind of data, unit, and protocol are used to represent this data item while the data source description provides this information. For instance, it specifies the data type is wind speed, the unit is m/s, and the protocol is REST[1]. In our system, data source descriptions are registered in the service registry and there is an address to get access to it which is stored in a specific repository.

- *Quality description:* is a description of the quality of data source. For example, the quality description of a data source determines that the completeness is 80% and the timeliness is 1 ms. Quality description is conceptually part of data source description but because of implementation simplicity we separate it from data source description.

[1] Representational State Transfer (REST) is a one of the emerging patterns of resource operations for the design of services in distributed systems such as Web 2.0 applications [41].

In this work, we discuss two types of data sources. The first one is real data source, which denote a regular data source connected to a real sensor. The second one is virtual data source, which is computed by combining one or more real data sources. There is an inevitable assumption for data source combinations: the data source description of the combining real data sources should be almost equivalent. Obviously, it is impossible to combine a wind speed data sources with a temperature data source. However, there are some cases where the combination becomes possible using a converter even though the real data sources are not completely equivalent. For example, if there are two wind speed data sources with different measurement units (e.g. m/s and km/h), it is possible to convert the units using a converter function before combining the two data sources.

Since the virtual data sources are generated by combining the real data sources, they also consist of three parts: data item, data source description, and quality description. The virtual data item is combined by a formula, e.g. average of data items from the real data sources. The virtual data source description is often remained the same to the real data sources or at least to one of the data sources. The quality description of the virtual data source is computed by combining the quality descriptions of the real data sources.

In some cases when we want to calculate the data quality of a data source, we need to compare the quality description of that data source with reality. But this is impractical because we are unable to measure the reality. One solution to tackle this problem is that we assume one of the real data sources with equivalent data source and quality description as the reality. This specific data source is called reference data source [29]. A reference (also known as benchmark in some literature [20]) is a data source that can be either a previous measurement of the same data source or mathematical model [42, 43] of that data .

2.2.2 Data Quality Dimensions

Many data quality dimensions have been proposed by various research projects. Table 2.1 shows an overview of data quality dimensions used in eight studies. The most commonly used quality dimensions are *accuracy*, *completeness*, and *timeliness*. Whereas, the dimensions such as *confidence*, *value-added*, and *coverage* are suggested by a couple of studies and the reasons for this are either these dimensions can be derived from the other dimensions or they might be applicable only in a few domains.

In this table some of the studies such as [30, 32] support several quality dimensions. These works are often general purpose data quality frameworks where they encompass a rich set of data quality dimensions. On the other hand, studies like [26, 29] only cover small number of dimensions because these works are applying data quality concept to a specific application domain and they only pick up the required dimensions in that domain.

We classify the quality dimensions into two groups: measurable and user interface. Measurable contains a set of dimensions that can be computed and used for data collection. On the other hand, the user interface group contains data presented to users in terms of diagrams or other tools. Typically, user interface is used to support data presentation. The two groups of data quality dimensions are as follows:

- **Measurable data quality dimensions:** accuracy, completeness, timeliness, consistency, access security, data volume.
- **User interface data quality dimensions:** relevancy, accessibility, confidence, coverage, objectivity, believability, reputation, value-added, interpretability, understandability, conciseness.

As we mentioned in this work we deal with sensor data and this can have different quality dimensions. So we pick up quality dimensions that are not only measurable, but also applicable in the wind energy domain. Another reason for this selection is that since we do not have data processing, the presentation of data is out of scope of this work.

Based on the studies that are listed in Table 2.1, we pick up main measurable data quality dimensions (i.e. *accuracy*, *completeness*, *timeliness*, and *consistency*). They are defined as follows:

Accuracy: we define accuracy as how close the observed data are to reality. According to ISO 5725 accuracy has two parts: precision and trueness [44] which are defined as follows.

- *Precision* is the closeness of agreement in individual results which is the standard deviation.
- *Trueness* is defined as the mean value of the difference of data source to the reality. We assume that the sensors are calibrated. Since we handle calibrated sensors, the

TABLE 2.1: Data quality dimensionts

	Classifications									
Dimensions	DeLone and McLean [33]	Wang and Strong [30]	Pipino et al. [31]	Lee et al. [32]	Bovee et al. [28]	Eppler [34]	Baumgartner et al. [29]	Geisler et al. [26]	Number of studies	**Measurable & User interface**
Accuracy	X	X	X	X	X	X	X	X	8	**M**
Completeness	X	X	X	X	X	X	X	X	8	**M**
Timeliness	X	X	X	X	X	X	X	X	8	**M**
Consistency		X	X	X	X	X	X	X	7	**M**
Access security		X	X	X		X			4	**M**
Data volume		X	X	X				X	4	**M**
Relevancy	X	X	X	X					4	**U**
Accessibility		X	X	X	X	X			5	**U**
Confidence							X	X	2	**U**
Coverage							X		1	**U**
Objectivity		X	X	X					3	**U**
Believability		X	X	X					3	**U**
Reputation		X	X	X					3	**U**
Value-added		X	X						2	**U**
Interpretability		X	X	X	X	X			5	**U**
Understandability	X	X	X	X					4	**U**
Conciseness	X	X	X	X		X			5	**U**

trueness is very close to the zero. Then we only use precision as the accuracy in our system.

Assume N is the total number of data points in data source D. The d_i is a data point of D and r_i is a reality value. The x_i is the difference between the data point and the reality. The μ denote the trueness. The accuracy of data source D is calculated Eq. (2.1):

$$\begin{aligned} &Accuracy(D) = \tfrac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2, \text{ Where} \\ &\mu = 0 \text{ and } x_i = d_i - r_i \text{ then} \\ &Accuracy(D) = \tfrac{1}{N}\sum_{i=1}^{N}(d_i - r_i)^2 \end{aligned} \tag{2.1}$$

The unit of accuracy is largely depends on the unit of the data source.

Completeness: is defined as the ratio of correctly received data points to the total number of sent data points. Let D denote a data source and R is the reference data source (reality). If the total number of sent data points in a given interval is N_R and N_D is the

number of data points in D, then the completeness data source D can be calculated by Eq. (2.2):

$$Completeness(D) = \frac{N_D}{N_R} \tag{2.2}$$

For example when there is a graph of wind speed data with 4000 data points and only 3200 of them are received the completeness of wind speed graph becomes 80%.

Additionally, the distribution of incompleteness could be varied. For example there is a difference between missing a data point every second and missing of all data points in one hour in the middle of the transmission.

Timeliness: is the average time difference between the producing of the data and the receiving of the data. Assume the t(r_i) denote the time when the reference data point i is produced and t(d_i) denote the time when the data point d_i is received. The timeliness of data source D is calculated Eq. (2.3):

$$Timeliness(D) = \frac{\sum_{i=0}^{N} t(d_i) - t(r_i)}{N} \tag{2.3}$$

Consistency: The extent to which the domain constraints have been met. Assume N_C is the number of consistent data points in data source D and N_R is the total number of data points in reference data source, then *Consistency(D)* represents the consistency of D and it is computed by Eq. (2.4):

$$Consistency(D) = \frac{N_C}{N_R} \tag{2.4}$$

For example, we define $S = \{x \in \mathbb{R} \mid 0 \leq x \leq 150\}$ is the set of valid data points for a wind speed data source with 4000 data points. So if we receive 3200 data points with values between zero and 150, the consistency of the D is 80%.

2.3 Chapter Summary

This chapter began with a review of exiting studies in two categories. The first category was data quality frameworks and applications where we introduced several data quality frameworks. The relation between these studies was discussed. The second category was

resource selection in SOA-based systems where we described different ways of selecting data sources in systems built based on SOA paradigm.

The second section of this chapter, dealt with defining the basic terms and concepts of this system. This part started with the definition of the data source. A data source consists of data item, data source description, and quality description. There are two types data sources: The real and virtual. A virtual data source is generated by combination of real data sources. The data quality dimensions and their classifications were introduced in the next part of this section. We classified 17 quality dimensions in to two groups: measurable and user interface. We picked up *accuracy*, *completeness*, *timeliness*, and *consistency* from the measurable group because they are relevant to our domain and also we only need computable dimensions. The chapter was closed by definitions, descriptions, and equations of selected quality dimensions.

Chapter 3

Towards A Data Quality-Based Framework

The previous chapter discussed the core components of the system and gave an overview of existing research. This chapter presents the design and core components of the proposed framework for quality-based resource management. It describes the interactions of all system components and their functionalities to enhance ESB with data quality handling feature.

3.1 Design Requirements

In order to develop scalable data quality-based applications, the proposed framework has to be general-purpose, and should not adhere to specific dependencies of any scenario. To achieve this requirement the design should allow for scalability with respect to the framework components, quality dimensions, and data source combination methodologies used.

Data quality dimensions have an important role in this approach and imply some design requirements. First of all, the framework must allow data providers to define the required quality dimensions. The user needs to acquire quality criteria from different data sources in a disciplined way.

Another important requirement is the independence of the framework from the ESB platform. The framework should attach and work to the ESB regardless of the platform type. However, in practice, some minor modifications are acceptable.

Finally, the framework should be independent of the target domain. Nevertheless, to be more realistic, a few changes are acceptable in different application domains.

3.2 Framework Architecture

In this section we describe the design of the framework by taking the predefined requirements into consideration. We propose a framework architecture, which uses a multilayer approach to handle data quality in ESB according to the previously set out requirements. The multilayer design, logically separates the functions and gives three main benefits: extendibility, high availability, and last but not least, performance. We describe them as follows [45]:

- Extendibility: this design allows the framework to employ different component technologies, ESB platforms, and deployment environments. It is possible to enhance the framework with new capabilities and components without having to make major modification to the framework.
- High availability: the use of multilayer ensures that the system can continue to work even though some of the components are down. For example, when some of the data provider servers are down, the system still can work out correctly and use the available providers to find the required data source.
- Performance: since the different layers can cache the requests from the other layers, the network utilization is minimized and the network load is reduced significantly.

Figure 3.1 illustrates an overview of the framework architecture. The three layers in the framework architecture carry out the task of resource management with respect to the quality descriptions of the resources.

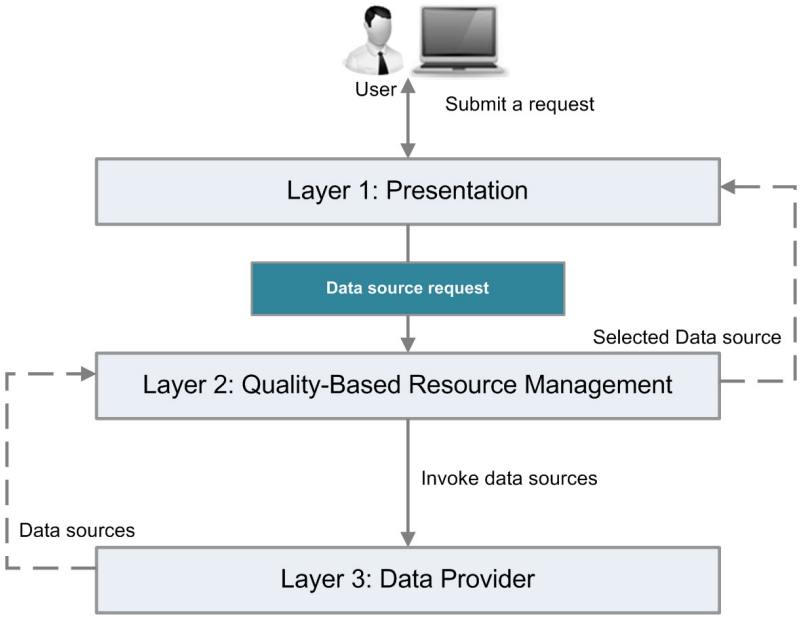

FIGURE 3.1: An overview of the proposed framework.

The first layer is the *Presentation layer*. It instantiates the submitted request from the user and if it is needed, modifies and passes the request to the second layer. The data source request is a formal description of the user's submitted request.

The next layer is the *Quality-Based Resource Management layer* and it selects the best data source with respect to the requested data, quality criteria, and constraints. This layer finds out the most appropriate data source for the user's request and gives that data source to the user.

The *Data Providers layer* involves a set of external data providers including the data, quality descriptions, and the way of access to the data. We will describe these three layers in detail later in this chapter.

Before we start to discuss the framework layers, we introduce the data source concept in the framework. In chapter 2, a high level definition of data source was given: a data source is one of the core components of our data quality-based system and it has three main parts: data item, data source description, and quality description. These parts are described in the context of the proposed framework as follows:

- *Data item:* is a stream of numbers coming from a sensor. The data item is a comma separated list of data point values (e.g. "10,14,13") that is generated by a service

deployed on a third party data provider. This part of the data source is only a collection of raw data that has not been subjected to any processing.

- *Data source description:* includes the information about the data source. In general, this meta data[1] is stored in the service registry. In order to access to this information, the system holds a reference to it.

- *Quality description:* is a list of quality dimensions with their values that can be assigned to the data item.

3.3 Presentation Layer

Figure 3.2 shows the structure of *Presentation layer* of the framework. The *Data source request* is an entry point to the data quality handling process. It defines the user's request precisely and is generated in four steps. The user starts with retrieving the domain information and quality dimensions from a *Domain repository*, for example, the user receives the list of available wind energy data together with the list of existing quality dimensions (Step 1 Fig. 3.2).

In the second step, the user selects the required semantic identification of data (i.e. name of data in our system), quality dimensions, constraints, and optionally one quality dimension for selection and optimization(Step 2), for example user selects wind speed with completeness of more than 80% and timeliness of less than 4 ms from May, 1 to the June, 1 with selection dimension of consistency. In the step three the selected semantic identification, quality dimensions, constraints, and optional selection dimension put together and the *Data source request* is generated (Step 3). When this request is forwarded to the layer 2, many processes are carried out by layer 2 and it ultimately returns results in response. In the last step, the results are given to the user (Step 4). The results are provided in two forms: the first form is a comma separated list of the data points. A graph displays these data point to the user. The second form is the subscription information. The users can use this information to subscribe to the selected data source and whenever a new data is published by the data source, all subscribed users will receive the published data.

[1]The meta data refers to the data that is describing another data [46].

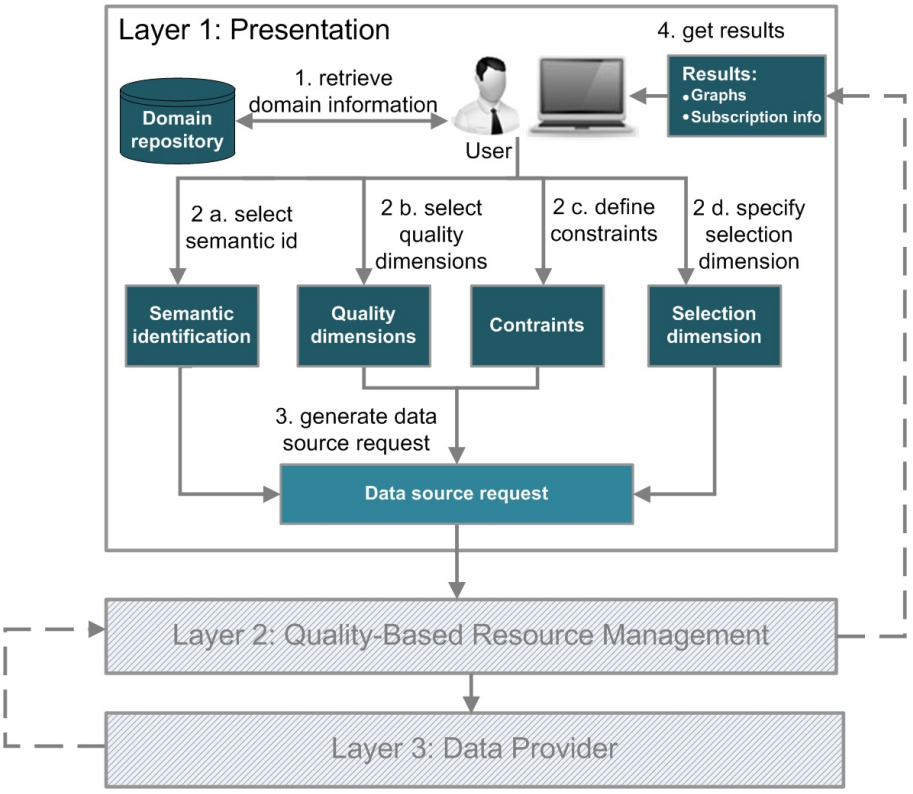

FIGURE 3.2: Presentation layer

The *Presentation layer* has the required interfaces (i.e. semantic id, quality dimensions, constraints, selection dimension) with *Quality-Based Resource Management layer* to communicate with layer 2. Consequently, there are two possible ways to design and implement *Presentation layer*: Firstly, using a User Interface (UI) that allows users to interact with system either graphically or on a command line basis [47]. Secondly, the programs that use the layer 2 interfaces to submit a request for a data source with specific quality conditions. Although we use a web-based UI for the sake of showing that our system works out correctly, it is possible to easily replace our UI with different types of programs such as console applications, form-based applications, and smartphone applications.

3.4 Quality-Based Resource Management Layer

The most important data quality processes are accomplished in this layer such quality calculation, quality-based selection, and quality combination. Thus, the *Quality-Based Resource Management layer* is the main focus of this thesis. The data quality handling process is split into three sub-layers *ESB*, *Service Manager*, and *Invocation Manager* as shown in Fig. 3.3.

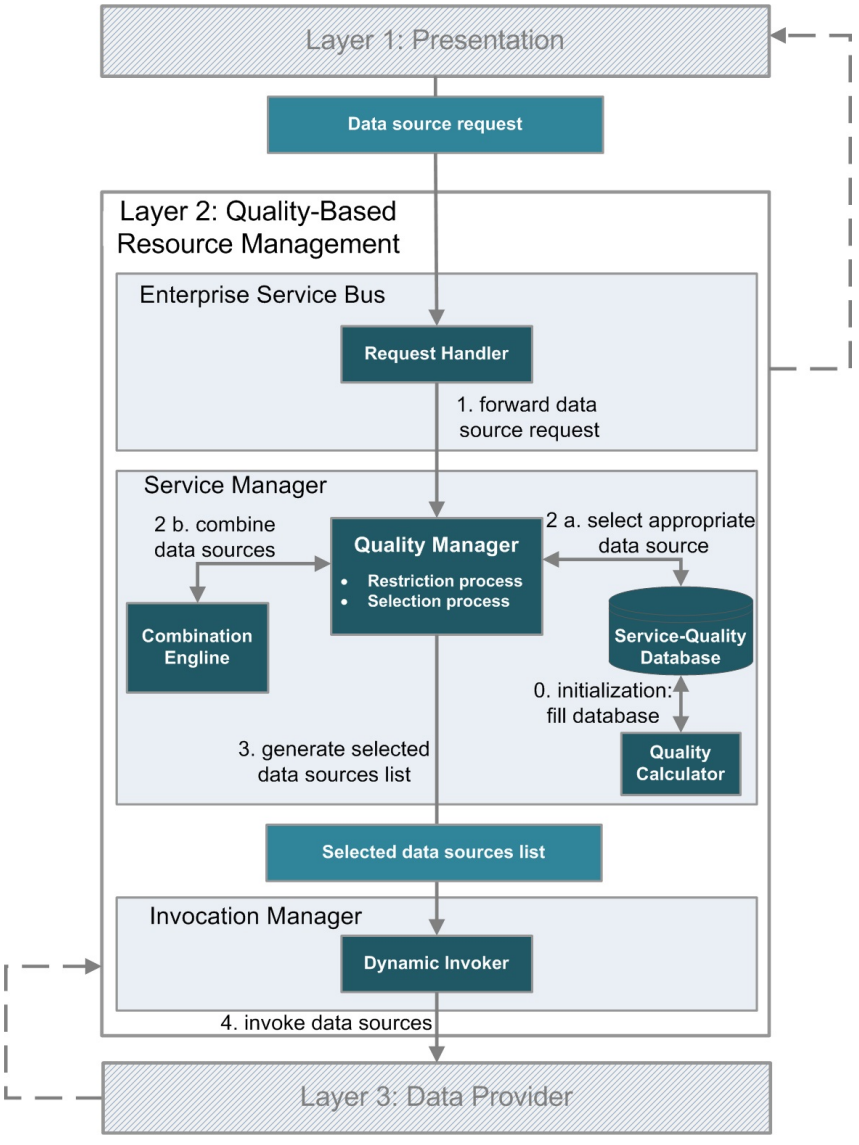

FIGURE 3.3: Quality-based resource management layer

The ESB contains one module that is called *Request Handler* to connect users to ESB and the other parts of the system. The *Request Handler* is the entry point to the ESB platform and responsible for providing connection between different components of the framework and the ESB. The only ESB platform specific component of the framework is *Request Handler*. Hence, by changing the ESB platform, the *Request Handler* would be replaced by a new module that is developed according to the target ESB platform specifications.

The *Request Handler* forwards the *Data source request* coming from layer 1 to the *Service Manager* (Step 1 - Fig. 3.3). The *Service Manager* sub-layer is in charge of finding proper data sources for the user and it contains four core components: *Quality Calculator*, *Quality Manager*, *Service-Quality Database*, and *Quality Combinator*.

At the beginning of the system and before the user starts sending requests (initialization), the *Quality Calculator* component calculates the quality descriptions of data sources according to the equations that have introduced in chapter 2. The *Quality Calculator* stores the calculated information into the *Service-Quality Database* (Step 0).

The *Quality Manager* selects the most appropriate data sources in terms of the data quality dimensions and constraints from the *Service-Quality Database* where all information about data sources, quality dimensions, and their corresponding values are stored (Step 2 a).

The *Quality Manager* uses two processes to find the best data source for the user: restriction process and selection process. The purpose of the restriction process is to detect and filter out outliers from the results according to the given quality dimensions and constraints. The restriction process prepares a list of available *good* data sources and hands in this list to the selection process. The basic idea of selection is to find the *best* data source among *good* data sources according to selection quality dimension. As we mentions in section 3.3, it is optional for the user to fill the selection quality dimension and the selection process would be skipped, if the user leaves this dimension empty. Therefore, if there are more than one data sources meet the user defined quality requirements, the *Quality Manager* select one the *good* data sources randomly.

Sometimes when the user sends a request for a data source with specific quality criteria, there is no data source available with the predefined quality requirements. To address this issue, the *Combination Engine* combines existing data sources and generates

a new data source that is called virtual (combined) data source (Step 2 b). The combination process includes combination of data items, data source descriptions, and quality descriptions. We will discuss the combination procedure in detail in section 3.6.

The *Quality Manager* generates the *Selected data source list* consists of the necessary information of the suitable data sources such as the data source address, parameters, and values. *Service Manager* passes the list to the next sub-layer (Step 3) which is *Invocation Manager*. By having the address of the data source, it is possible to get access to the data source. The data sources are implemented as services in our system and services should be invoked to get access to their data. The *Dynamic Invoker* reads the data sources in the *Selected data sources list* and invokes them dynamically.

The idea of three-layered approach has been introduced for two reasons. Firstly, by splitting the tasks of data quality handling process into separate layers it enhances the extendibility and scalability of the framework. Secondly, having a separate ESB layer, isolate this platform from other parts of the framework. Consequently, the independence of the framework from ESB platform guaranteed and the ESB platform replacement becomes a simple and light weight process. As a result, a wider number of users can take advantages of our framework regardless of their ESB technology.

3.5 Data Providers Layer

Figure 3.4 shows the third layer of the framework which is the *Data Providers layer* and represents a set of external data sources. As we have described a data source has three parts a data item, service description and a quality description. The data item provides the data in the terms of services with local databases of sensors data. These services are accessible through their service descriptions. The *Dynamic Invoker* binds and invokes the data item of the selected data sources in its list (Step 1 Fig. 3.4). When the service is invoked, it executes a query on its local *Data source database* and retrieves the stored data points in database (Step 2).

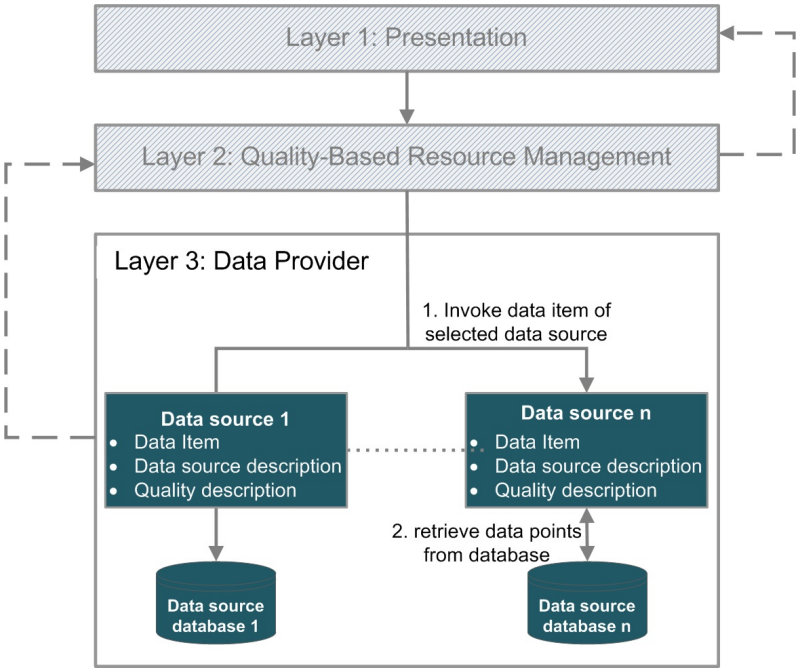

FIGURE 3.4: Data providers layer

3.6 Data Quality Combination

In order to improve the data quality, it is possible to combine the different data sources. This section shows how to combine quality dimensions by using three combination methods as follows:

- *D1 (A) D2:* the first combination method is a conventional average of the data sources D1 and D2.
- *D1* $\bigoplus$ *D2:* this combination method is defined as, first pick up the data points from data source D1 if available otherwise, pick up from the D2.
- *D1 (E) D2:* is defined as, pick up the earliest receiving data point regardless of the fact that the data is belong to D1 or D2.

3.6.1 D1 (A) D2

Completeness: Let Compl(D1) denote the probability of the event, *"D1 having data available"* and Compl(D2) denote the probability of the event, *"D2 having data available"*.

We assume $\overline{Compl(X)} = 1 - Compl(X)$. The completeness of the virtual data source is calculated by Eq. (3.1):

$$Compl(D1\ (A)\ D2) = \overline{\overline{Compl(D1)}\ .\ \overline{Compl(D2)}} \tag{3.1}$$

Accuracy:
Assumption 1: The accuracy of the D1 and D2 are two random variables with normal distribution.
Assumption 2: D1 and D2 are independent sources.
We defined accuracy as the distribution of difference to the reality. So let True(D1) and True(D2) are trueness of D1 and D2 respectively. Then the tureness of the virtual data source is shown by Eq. (3.2):

$$True(D1\ (A)\ D2) = \frac{True(D1) + True(D2)}{2} \tag{3.2}$$

However, the because we assume that the sensors are calibrated the tureness is often considered as zero. Then the accuracy is equal to the precision. Let Preci(D1) and Preci(D2) are precision of D1 and D2 respectively. The precision of the virtual data source is shown by Eq. (3.3):

$$Preci(D1\ (A)\ D2) = \frac{Preci(D1) + Preci(D2)}{4} \tag{3.3}$$

By using the above equation, when the accuracy of D1 and D2 are far from each other the accuracy of the combined data source can not be better than both and then the virtual data source is not selected. On the other hand, when the accuracy of two data sources D1 and D2 are close to each other the accuracy of virtual data source can better than both. For example assume the accuracy of D1 and D2 are 10 m/s and 3 m/s, by using Eq. (3.3) the accuracy of the virtual data source becomes 5.22 m/s which is not better than D2 (We know that the lower accuracy value is the better because we assume accuracy as difference from reality). In this case D2 will be selected. But if D1 and D2 have the accuracies of 10 m/s and 8 m/s respectively, the accuracy of virtual data source becomes 6.4 m/s with is better than both D1 and D2 and then the virtual data source will be selected for the user.

Timeliness: Assumption 1: The timeliness of the D1 and D2 are two random variables with same exponential distribution.

Assumption 2: D1 and D2 are independent sources.
Let Time(D1) and Time(D2) denote the timeliness of D1 and D2 respectively. The timeliness of virtual data source by Eq. (3.4):

$$Time(D1\ (A)\ D2) = \frac{3Time(D1)}{2} \tag{3.4}$$

The timeliness in this case needs more detailed elaborations to calculate and probably it needs more parameters but this is not our concern.

3.6.2 D1 $\bigoplus$ D2 method

Completeness: Let Compl(D1) denote the probability of the event, *"D1 having data available"* and Compl(D2) denote the probability of the event, *"D2 having data available"*, then the completeness of the virtual data source is calculated by Eq. (3.5):

$$Compl(D1 \bigoplus D2) = \overline{\overline{Compl(D1)}\ .\ \overline{Compl(D2)}} \tag{3.5}$$

Accuracy: Assumption 1: The accuracy of the D1 and D2 are two random variables with normal distribution.
Assumption 2: D1 and D2 are independent sources.
We defined accuracy as the distribution of difference to the reality. Let Compl(D1) denote the probability of the event, *"D1 having data available"* and Compl(D2) denote the probability of the event, *"D2 having data available"*. In addition we assume True(D1) and True(D2) as trueness of D1 and D2 respectively. Then the tureness of the virtual data source is shown by Eq. (3.6):

$$True(D1 \bigoplus D2) = \frac{Compl(D1) * True(D1) + \overline{Compl(D1)} * Compl(D2) * True(D2)}{Compl(D1) + \overline{Compl(D1)} * Compl(D2)} \tag{3.6}$$

However, the because we calibrate the sensors the tureness is often considered as zero. The trueness is not in *Quality-Service Database* although, if it is needed we can calculate it with the above equation.

The precision is considered as the accuracy. Let Preci(D1) and Preci(D2) are precisions of D1 and D2. The precision of virtual data source is calculated by Eq. (3.7):

$$Preci(D1 \bigoplus D2) = \frac{Compl(D1) * Preci(D1) + \overline{Compl(D1)} * Compl(D2) * Preci(D2)}{Compl(D1) + \overline{Compl(D1)} * Compl(D2)} \tag{3.7}$$

Timeliness: Let Time(D1) and Time(D2) denote timeliness of D1 and D2 respectively. Then the timeliness of virtual data source is calculated by Eq. (3.8):

$$Time(D1 \bigoplus D2) = \frac{Compl(D1) * Time(D1) + \overline{Compl(D1)} * Compl(D2) * Time(D2)}{Compl(D1) + \overline{Compl(D1)} * Compl(D2)} \tag{3.8}$$

3.6.3 D1 (E) D2 method

Completeness: Let Compl(D1) denote the probability of the event, *"D1 having data available"* and Compl(D2) denote the probability of the event, *"D2 having data available"*, then the completeness of the virtual data source is calculated by Eq. (3.9):

$$Compl(D1\ (E)\ D2) = \overline{\overline{Compl(D1)}\ .\ \overline{Compl(D2)}} \tag{3.9}$$

Accuracy:
Let Acc(D1) and Acc(D2) are accuracy of D1 and D2. The α is the as the weight for the accuracy. The accuracy of virtual data source is calculated by Eq. (3.10):

$$Acc(D1\ (E)\ D2) = \alpha Acc(D1) + \overline{\alpha} Acc(D2) \tag{3.10}$$

By having the α the calculation of accuracy is possible.

Timeliness: Assumption 1: The timeliness of the D1 and D2 are two random variables with same exponential distribution.
Assumption 2: D1 and D2 are independent sources.
Let Time(D1) and Time(D2) denote the timeliness of D1 and D2 respectively. The timeliness of virtual data source by Eq. (3.11):

$$Time(D1\ (E)\ D2) = \frac{Time(D1)}{2} \tag{3.11}$$

The timeliness needs more detailed elaborations to calculate and probably it needs more parameters but this is not our concern.

The Table 3.1 gives an overview of all combination methods with data quality dimensions. These methods are used to generate the virtual data source from the real data sources.

TABLE 3.1: Data quality dimensions and combination methods

	Data quality dimensions		
Method	Completeness	Accuracy	Timeliness
D1(A)D2	$\overline{\overline{C(D1)} \,.\, \overline{C(D2)}}$	$\frac{P(D1)+P(D2)}{4}$	$\frac{3Time(D1)}{2}$
D1$\bigoplus$D2	$\overline{\overline{C(D1)} \,.\, \overline{C(D2)}}$	$\frac{C(D1)*P(D1)+\overline{C(D1)}*C(D2)*P(D2)}{C(D1)+\overline{C(D1)}*C(D2)}$	$\frac{C(D1)*T(D1)+\overline{C(D1)}*C(D2)*T(D2)}{C(D1)+\overline{C(D1)}*C(D2)}$
D1(E)D2	$\overline{\overline{C(D1)} \,.\, \overline{C(D2)}}$	$\alpha Acc(D1) + \overline{\alpha} Acc(D2)$	$\frac{Time(D1)}{2}$

By combining data quality dimensions, we aim to generate a virtual data source with better data quality descriptions. The three combination methods have different effect on data quality dimensions. A method can increase or decline the quality. Sometimes a method can increase or decline the data quality depends on the receiving data. The Table 3.2 shows relation between the combination operations and data quality dimension.

TABLE 3.2: Quality Combination Relations

	Data quality dimensions		
Combination method	Completeness	Accuracy	Timeliness
D1 (A) D2.	✓	✓	×
D1 $\bigoplus$ D2	✓	−	−
D1 (E) D2	✓	−	✓

✓: It can be better than both of the D1 and D2[1].

−: It is probably between the D1 and D2.

×: It is worth than both D1 an D2.

According to this table, all three methods can increase the completeness. Consequently, when the completeness is the priority of the user, the combination is strongly suggested with one of these methods. Using average method the combined data source would have better accuracy. On the other hand, the average not only is not successful to increase the timeliness, but also makes it worse in any situation. Therefore, if the timeliness is a critical factor from user point of view, the average is not suggested. For the $\bigoplus$ method both accuracy and timeliness the combined data source probably has a accuracy and timeliness between D1 and D2. Therefore, this table suggest that the $\bigoplus$ does not used. The table also suggest the (E) method for timeliness because it would be better than both D1 and D2.

3.7 Chapter Summary

This chapter has explained the proposed framework architecture for quality handling in ESB. The data quality framework uses a multi-layered patten to isolate and separate functionalities of each layer and the quality handling process effectively. There are three layers in the system: *Presentation layer*, *Quality-Based Resource Management*, and *Data Provider layer*.

The chapter discusses how a user request's is handled by a collection of servers called integration server and the most appropriate data source is selected for the user. The

[1]One example for this claim is as follows: we assume the completeness of 80 % and 70 % for D1 and D2. By putting these values in Eq. (3.1) the accuracy of D1 (A) D2 becomes 94 %.

chapter ends by a discussion about data quality combination and different combination strategies.

Chapter 4

Implementation

This chapter discusses the internal implementation details of the framework. The initial prototype contains three main parts, web-based client application, integration server, and data provider services. Figure 4.1 shows these components and their interactions. The web-based client application receives the request from the user and forwards it to the integration server. This application also shows the data source one a graph. The integration server is responsible for data quality handling and communicating with data providers. The data providers store the data and provides an address to access that data. These three main components of the system are described as follows:

4.1 Web-based Client Application

The web-based client application consists of two parts: the first is an UI that is used to send a request. The second part is a graph to show the results to the user.

The UI gives the possibility to choose the name of the data, the quality dimensions, the constraints, and one quality dimension as selection dimension. Figure 4.2 shows the user select the wind speed as the data name, and completeness of more than 80 % and timeliness of less than 4 ms as quality dimension and constraints. The optional selection dimension is the consistency.

When the user sends this request the integration server processes the request and responses with a data source. The graph of the selected data source is shown to the user. The UI application uses Ajax to bind to the ESB. Ajax is a set of interrelated technologies that simplify the creation of asynchronous web applications. Ajax enables

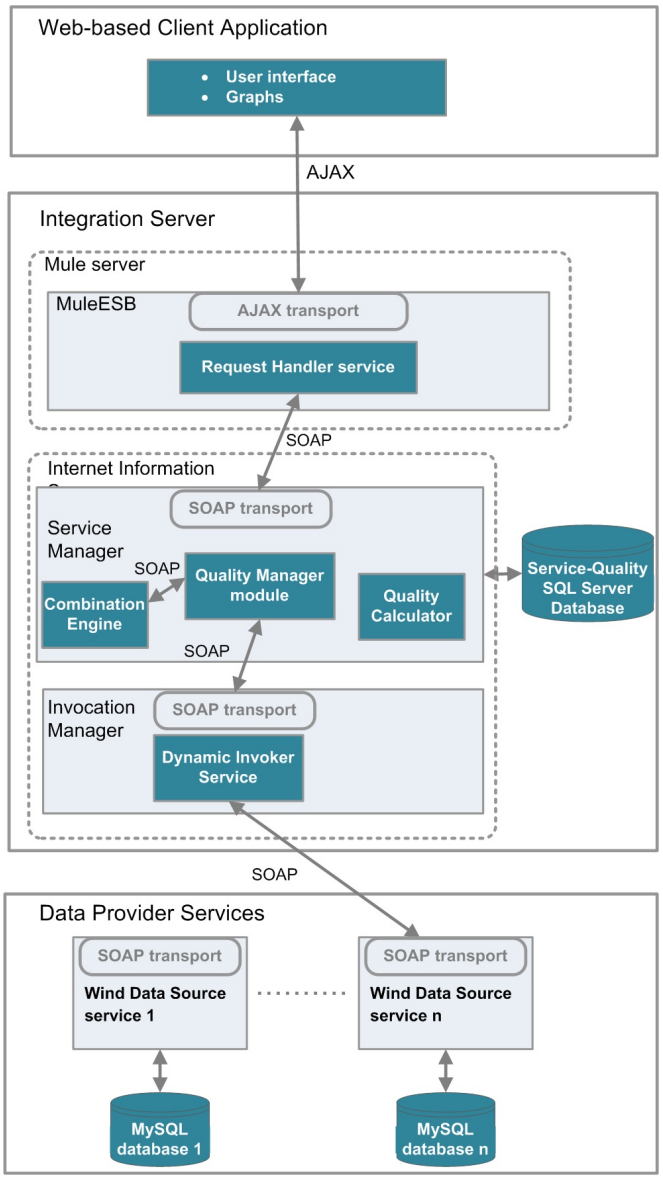

FIGURE 4.1: Prototype implementation of the framework

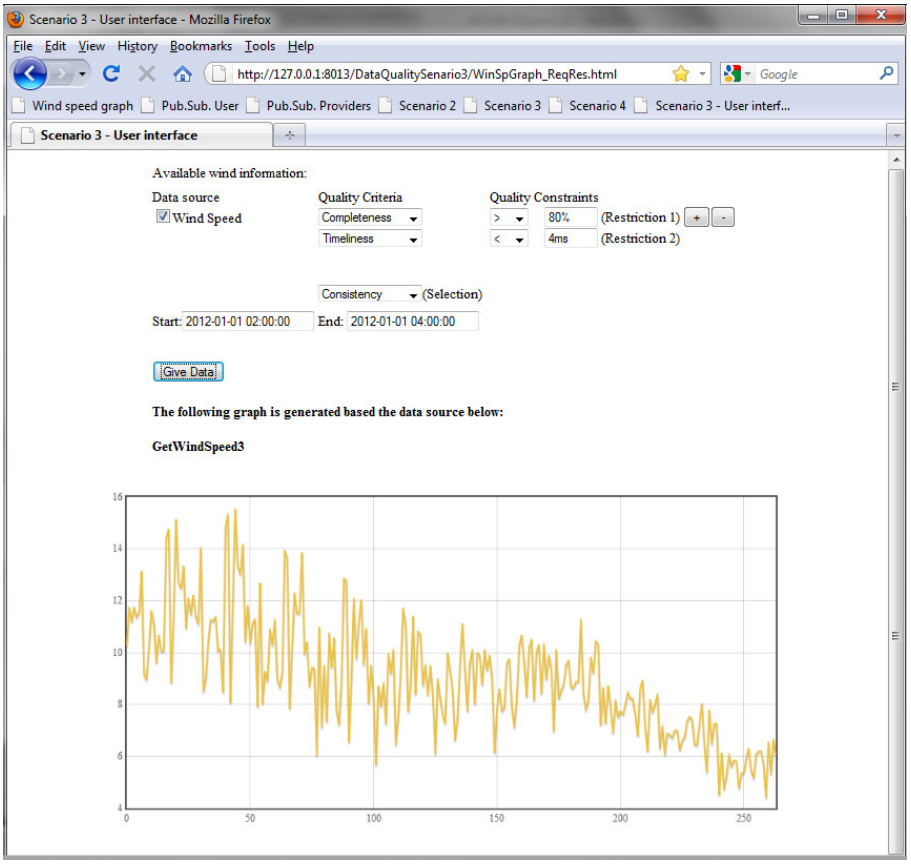

FIGURE 4.2: User can choose the required data name, quality dimension, constraint, and selection dimension

the web applications are to send and receive data without refreshing the existing web page [48]. We use Flot to produces graphs on a web browser. Flot is a Javascript plotting library for jQuery and is used to provide graphical plots of datasets on-the-fly at the client side [1].

4.2 Integration Server

The integration server consists of following main components:

- MuleESB[2]: is an open source enterprise service bus framework and we use it as the communication backbone of the system because of its simplicity, large number

[1] https://code.google.com/p/flot/
[2] http://www.mulesoft.org

of connectors, transports, and its light weight foot prints [38]. MuleESB can host several Mule applications. The Mule application executes one or more Mule flows in the form of Extensible Markup Language (XML). A Mule flow consists of pre-packaged building blocks which are defined in specific sequences. Mule application processes and orchestrates Mule messages based on those sequences. A Mule message crosses from one block to the next block in the Mule configuration file while each block processes the message and react according to its configuration. MuleESB provides Ajax namespace and Ajax connector in order to bind Mule flow and web services to the Ajax channel on the browser. The Ajax endpoint is configured as an outbound operation. It creates a transport channel to send messages asynchronously to and from an Ajax server, which connects Mule flow to an external web page. A JavaScript function attached to the web-based client application listens for incoming messages. It extracts and classifies the received data and shows it on the web page. Figure 4.3 show mappings of a flow to its corresponding XML configurations in MuleESB for handling of wind speed.

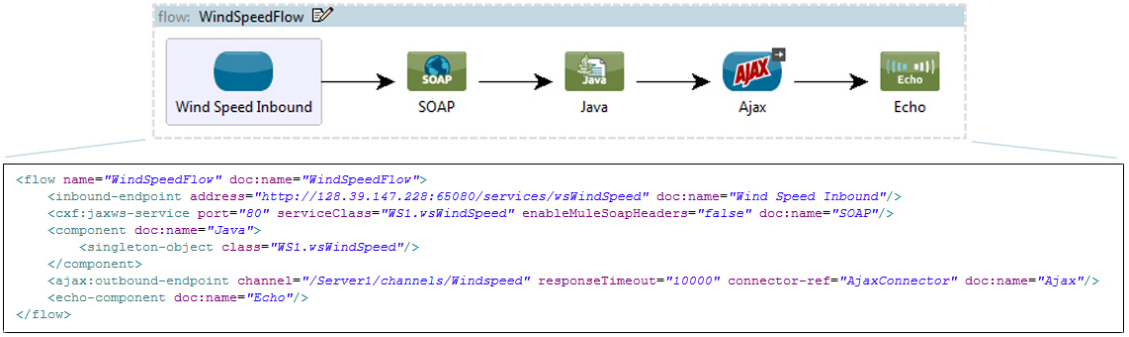

FIGURE 4.3: A Mule ESB flow for wind speed.

- Service Manager: is a .net web service and that is deployed on Internet Information Services (IIS) [1] web server. It contains a Quality Manager module which is responsible for analysing the data quality. The Quality Manager finds the best data source information and asks the Dynamic Invoker to invoke the required data.
- Dynamic Invoker: is also a .net web service that dynamically binds and invokes to the selected data sources.

[1] http://www.iis.net

- Service-Quality Database: is a SQL Server[1] relational database that stores information about all entities of the system in two main categories: service and quality information. The first category is used to store information of services and invoke them. The second category stores information required for data quality handling. Quality Manager executes query on quality information category for selecting the required data item and the result of query typically is a list of Web Service Definition Language (WSDL)[2] address, required operations, and their input/output parameters. These lists are delivered to the Dynamic Invoker Service for subsequent biding and invocation. The Figure. 4.4 show the Service-Quality Database scheme when the tables are classified two aforementioned categories. ..

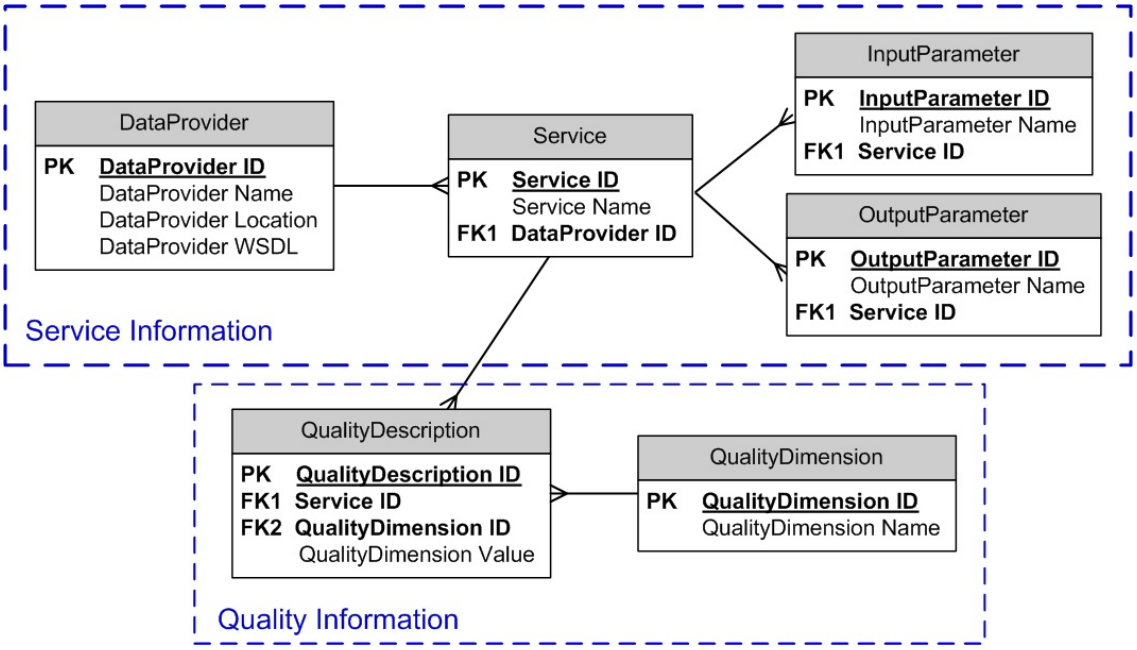

FIGURE 4.4: The database Entity Relationship Diagrams (ERD) of Service-Quality database

- Request handler: is a Java web service deployed on the MuleESB. This service takes the information from web-based client application and forwards it to Service Manager.

[1] http://www.microsoft.com/en-us/sqlserver/default.aspx

[2] WSDL is an XML-based language that provides a description of a web service's functionalities. The WSDL file is accessed by a WSDL address.

- Quality calculator: The quality calculator is .net web service which has a separate function for each quality dimension. It often takes a data source and a reference data source and computes the quality value. After computing the quality dimension the quality calculator can store the information in the Service-Quality Database.
- Combination Engine: is a .net web service that has a function for each dimension. It takes a list of data sources and combines them according to the corresponding combination formula and store a new virtual data source in the Service-Quality Database.

4.3 Data Providers

Data provider services are third party services that provide data items, data source descriptions, and quality information of the data. We use Java API for XML Web Services (JAX-WS)[1] client to send data and quality using SOAP messages. A data provider announces the quality descriptions by publishing the quality descriptions as an extra service. This service is characterized by a suffix of "_Quality". For example, Get_WindSpeed service has a corresponding quality description service that is called Get_WindSpeed_Quality. The data providers contain third party services and local databases. They enable the access to the data that are stored in the databases by means of a set of services.

4.4 Chapter Summary

In this chapter we describe the prototype implementation of the proposed framework. According to the design of the framework the implementation also has three layers. We describe implementation details of main components in the system. First we describe the web-based client application. Then we explain the implementation of integration server consists of several important components such as MuleESB, Server Manager, Quality Manager and so on. Finally, we present the implementation of the data providers.

[1] https://jax-ws.java.net/

Chapter 5

Evaluation and Results

5.1 Scenarios

This chapter introduces five usage scenarios in the scope of the offshore wind domain to show how the data quality can be handled in ESB using the proposed framework. One important component of the offshore wind management systems is the windmill. A number of wind sensors (e.g., wind speed sensor) are attached to a windmill [49]. As offshore windmills are located far from the coastline where the weather is harsh, sensors are subject to the moisture and corrosion [22]. Therefore, the quality of the data produced by them can be negatively influenced [50].

A data quality-based resource management system in offshore wind energy domain requires measurements of different wind information from wind sensors. Each sensor can send the measurements to the integration server through a third party wind service. A user can send a request to the sensors data via the integration server. The integration server performs the data quality handling process and responses to the user. The descriptions of the implemented scenarios are as follows:

5.1.1 Scenario 1

Data sources: There are three real wind speed data sources in the system.

Data quality dimensions: There is only one quality dimension (i.e. completeness) for each data source.

User's request: The user issues a request for wind speed with the completeness of more than 85%.

Fig. 5.1 shows the scenario 1 where the data is mainly provided by wind sensors through a number of wind services.

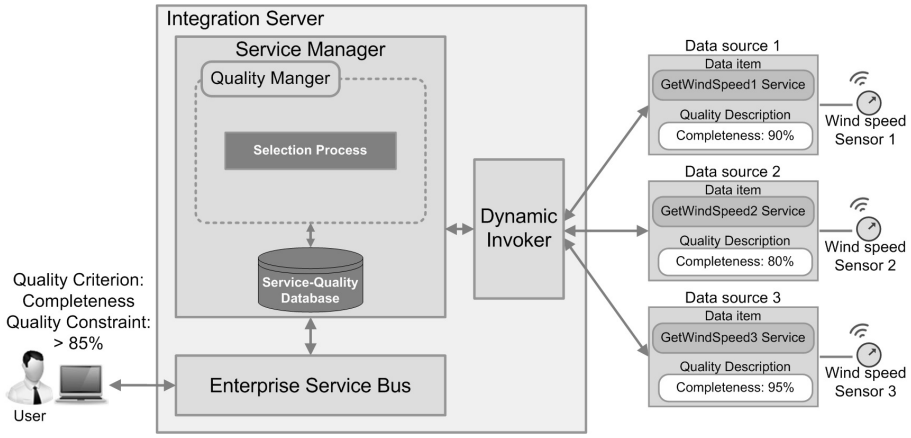

FIGURE 5.1: Scenario 1. Data sources: three real data sources. Quality dimension: completeness. Question: completeness of more than 85%.

Results: There are two data sources with completeness of more than 85%: data source 1 and 3, they have the completeness of 90% and 95% respectively. Since the user does not specify the selection dimension, the *Quality Manger* can select either data source 1 or 3 arbitrarily. Figure 5.2 shows the graph of data source 1 that is selected by *Quality Manger* and is shown to the user.

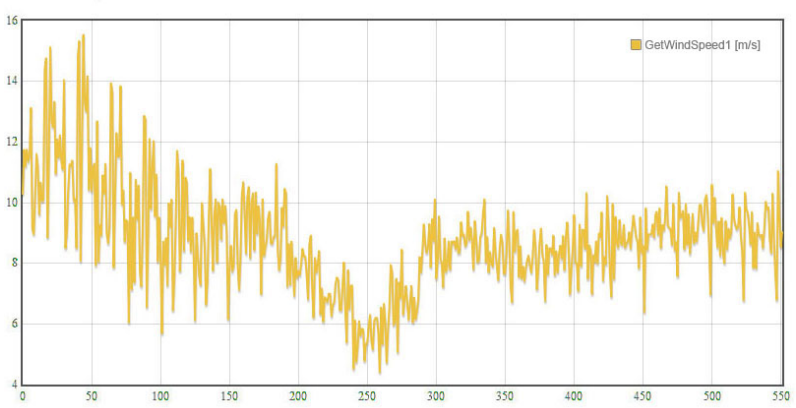

FIGURE 5.2: The data source 1 is selected and its graph is shown to the user.

5.1.2 Scenario 2

Data sources: There are two real wind speed data sources and one virtual data source that is derived from the two real data sources.

Data quality dimensions: There is only one quality dimension (i.e. completeness) for each data source.

User's request: The user issues a request for wind speed with completeness of more than 85%.

Fig. 5.3 shows the scenario 2 where the data is mainly provided by wind sensors through a number of wind services.

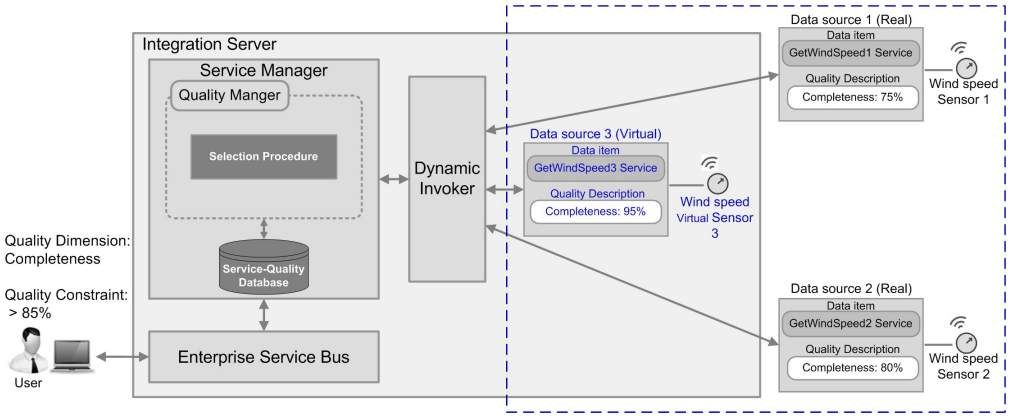

FIGURE 5.3: Scenario 2. Data sources: one virtual and two real data sources. Quality dimension: completeness. Question: completeness of more than 85%.

Results: The completeness of data sources 1, 2, and 3 are 75%, 80%, and 95% respectively which means by combining two real data sources a virtual data source with higher completeness is generated. The reason is that each of the data sources 1 and 2 has missed some parts of the data but when they are composed, each data source might fills the missing part of the other one. Therefore, the virtual data source that is selected by *Quality Manger* and its graph is shown to the user.

5.1.3 Scenario 3

Data sources: There are three real wind speed data sources in the system.

Data quality dimensions: There are two quality dimensions (i.e. completeness and timeliness) for each data source.

User's request: The user issues a request for wind speed with two types of quality dimensions: the first is completeness of more than 75% which is used for the restriction process. In addition, user chooses the timeliness as the selection quality dimension.

Fig. 5.4 shows the scenario 3 where the data is mainly provided by wind sensors through a number of wind services.

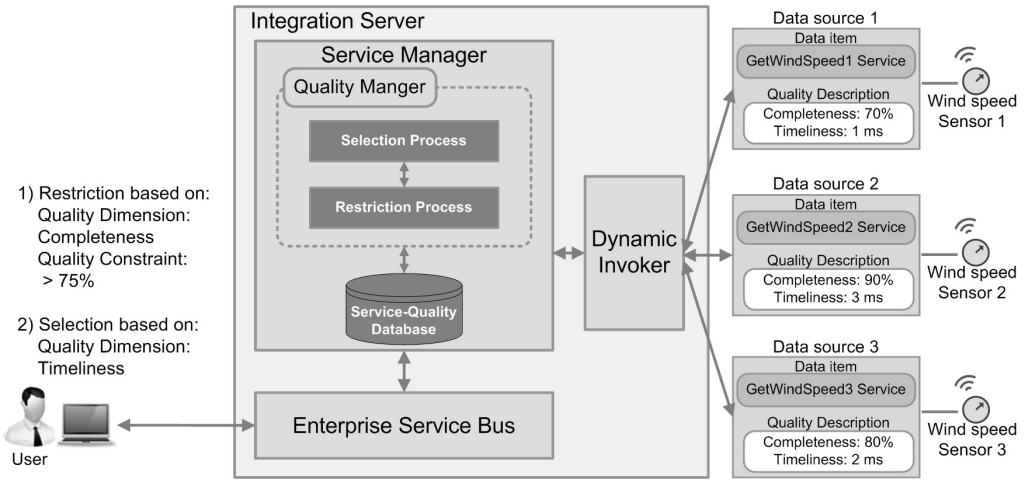

FIGURE 5.4: Scenario 3. Data sources: three real data sources. Quality dimensions: completeness and timeliness. Question: completeness of more than 75% and selection quality dimension is timeliness.

Results: The two data sources 2 and 3 have the completeness of more than 75% (the restriction process) and they are considered as *good* data sources. Since the data source 3 has better timeliness (i.e. 2 ms), the *Quality Manager* select the data source 3 as the *best* data source (selection process) and its graph is shown to the user. In this scenario the user prefers to get data source as soon as possible with a reasonable completeness. The data source 1 is too incomplete and filtered by restriction process. From the two remaining data sources the data source 2 has the higher completeness while the data source 3 has better timeliness. The data source 3 is selected because the user's priority is the timeliness and a few incompleteness of data is acceptable.

5.1.4 Computation of data quality scenario

Data sources: There are three real wind speed data sources in the system. One of them is considered as the reference data source. The quality descriptions of the data sources 1 and 2 are not available.

Data quality dimensions: There is only one quality dimension (i.e. completeness) for the reference data source.

User's request: The user issues a request for wind speed with the completeness of more than 85%.

Figure 5.5 shows the communications between different components of the system in this scenario.

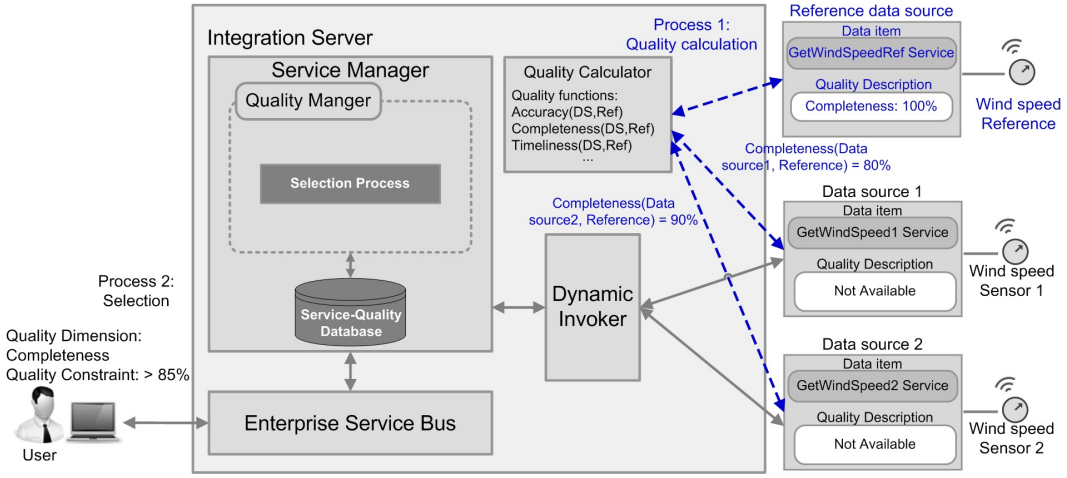

FIGURE 5.5: Computation of data quality scenario. Data sources: one reference and two data sources without quality description available. Quality dimension: completeness. Question: completeness of more than 85%.

Results: By using the reference, the *Quality Calculator* computes the completeness of 80% and 90% for data source 1 and 2 respectively and stores these values in *Service-Quality Database*(Process 1). The data source 2 is selected for the user (Process 2).

5.1.5 Publish/subscribe scenario

Data sources: There are three real wind speed data sources in the system that are pushing the data.

Data quality dimensions: There is only one quality dimension (i.e. completeness). The data sources 1, 2, and 3 have the completeness of 65%, 80%, and 70% respectively.

User's request: The user issues a request for subscribing to a wind speed data source with the completeness of more than 75%.

In the publish/subscribe pattern, there are some publishers and subscribers in the system. Subscribers ask for subscription to the publishers and whenever a publisher publishes any data item, all of its subscribers would receive the published data items [51].

Results: In this scenario when a user sends a request the *Quality Manger* finds the best data source (i.e. data source 2) to be subscribed, and gives the subscription information to the user. This information is an ESB channel name (i.e. "/Channels/GetWindSpeed2"). When user receives this channel name, it subscribes to the channel by calling a function in the MuleESB called Mule.Subscribe("channelname"). Once the user subscribes to the channel he can receive all data that are pushed by the data providers. Basically in publish/subscribe pattern scenario the *Dynamic Invoker* component is not used, because the data sources are no more invoked.

Figure 5.6 shows a web-based application that displays the data providers with their data sources descriptions (such as type of data, wind farm location, and so on) together with their quality descriptions. They are pushing the data to the ESB and then there exists three channels for users to be subscribed.

FIGURE 5.6: Publish/subscribe scenario. Data sources: three real data sources that are pushing the data. Quality dimension: completeness. Question: completeness of more than 75%.

Fig. 5.7 illustrates once the user is subscribed to a data source 2 using the "/Channels/GetWindSpeed2", he receives all real-time data coming from data source 2.

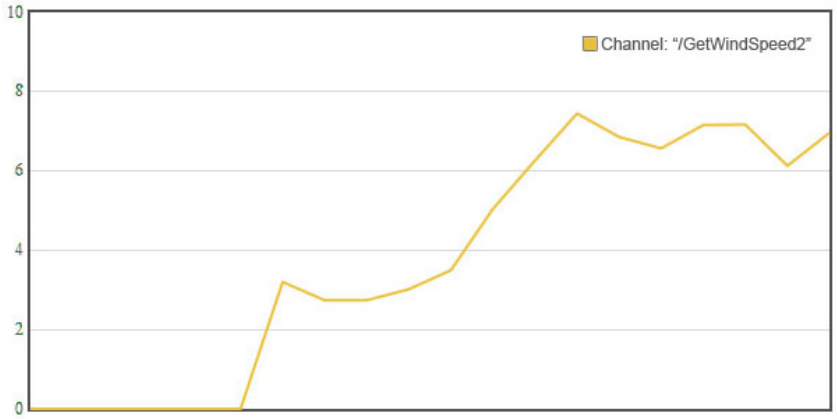

FIGURE 5.7: Publish/subscribe scenario. The user has subscribed to the data source 2 and is receiving the real-time data.

As the publish/subscribe pattern is a specific messaging model supported by our framework, this model can be used in all previously described scenarios to receive the real-time data instead of data stored in the database.

5.2 Chapter Summary

In this chapter we describe five usage scenarios to show the validity of our data quality-based framework. These scenarios have been implemented in offshore wind energy domain. The first scenario shows the data quality selection processes using to one data quality dimension. The data quality combination process is described in the scenario 2 where there is one virtual data source derived from the two real data sources. The third scenario evaluates the framework when multiple data quality dimensions are in the system and the user can choose some of them for restriction process and one of them for selection process.

The computation of data quality scenario and publish/subscribe scenario are basically different from the first three scenarios. These scenarios discuss two general issues that can be applied to three other scenarios. In the computation of data quality scenario, there are two data sources without data quality descriptions available. The quality calculator employs a reference data source to compute the quality descriptions of data sources. The chapter ends by the publish/subscribe scenario. In this scenario, the user sends a

request for a real-time data. The quality manager responds with data source subscription information rather than the data source service address.

In this chapter we present how each of the requirements for data quality-based resource management, identified in section 1.2 is met. Base on the results obtained by aforementioned scenarios with different configurations and assumptions, the proposed framework in this thesis can be considered as a robust framework for data quality handling in ESB.

Chapter 6

Conclusion and Future Work

6.1 Conclusions

As the ESB is gaining more attention from industry, there is an increasing need to specify ESB's resource management in a disciplined manner. The proposed framework presents a novel data quality-based resource management model for ESB.

In traditional approaches user needs to manage the quality descriptions directly and knows lots of implementation details in the data providers and their quality specifications. The framework moves the data quality process from the user level to the middleware level (i.e. integration server).

By preventing the users from dealing with the data providers, the data quality process made easier. This lifts the user's experience in the sense of decreasing the details that user has to know in order to get access to data sources and their quality information. This framework makes an abstraction that is a data model and the user is just relating to the domain description. The user starts with the domain description and asks a data source with specific quality criteria and constraints, and then the framework selects the most suitable data provider for the user. The framework presents a way to data quality combination that leads to notable improvement in data quality descriptions. The framework introduces a general-purpose solution for describing, measuring, and comparing the data quality descriptions in ESB.

6.2 Contributions

This thesis makes three major contributions as follows:

1. The thesis reviews existing research in the data quality and service selection areas and presents main research challenges that need to be solved by a new approach to facilitate resource management in ESB.

2. It explains the architecture and implementation of a novel quality-based framework for ESB applications. It identifies the mechanisms and techniques that the system is relay on to work efficiently.

3. It presents handling of data sources and their quality information. A new approach for data quality combination/calculation is proposed and developed.

6.3 Future Work

The result of this research raises a number of potential directions for the future work as follows:

One interesting direction for extending the proposed framework is to enhance it with semantic technologies. A semantic-enhanced data quality framework is obtained by replacing the *Service-Quality Database* with ontology. The ontology is a precise representation of the knowledge with a number of concepts and their relationships within a specific domain [52]. In a *Service-Quality Database*, the concepts are stored using tables, but the framework is unable to know about what the meaning of the concepts is and how these concepts are associated with each other. On the other hand, the ontologies can provide a way to store such information. By replacing the ontology, the framework enhanced with more advanced and intelligent data quality selection queries. It introduces a new level of abstraction that can improve the process of quality management defined by using semantic technologies.

One extension of this framework could be the improvement and optimization in communication style. We employ the SOAP-based message exchange for communication between internal components of the framework as well as communication between the framework and external data providers and users. As we mentioned, one emerging information access style is REST. There is an ongoing debate about whether SOAP should be replaced by REST or not [53, 54]. To support the users and data providers who are

using REST, the data quality-based framework should be enhanced with REST message exchange. We can replace the internal communication with REST while for external communications, we can add the REST as a new feature to support both SOAP and REST users.

With the growing number of alternative data sources that can be combined to generate virtual data sources, the data quality combination becomes a decision problem. To achieve higher data quality, we have to know which data sources should be selected for combination and how they should combined. In order to solve these problems the process of selecting data sources for combination has to be optimized. Therefore, the data quality combination optimization can be considered as another future work. One possible solution is use of a tree-structured data source category. In such a system, the data sources and quality descriptions are hierarchically classified into several combination zones. Each node of that tree includes a number of data sources with strong possibility of successful combination. In addition, each node consists of a combination strategy and the whole tree forms a data quality combination plan. This combination plan includes sequences of combinations that can be executed by the framework to generate virtual data sources with higher data quality descriptions.

Bibliography

[1] Douglas K Barry. *Web Services and Service-Oriented Architectures: The Savvy Manager's Guide*. Morgan Kaufmann, 1st edition, April 2003.

[2] Thomas Erl. *Service-oriented Architecture: Concepts, Technology, and Design*. Pearson Education India, 2006.

[3] Martin Keen, Amit Acharya, Susan Bishop, Alan Hopkins, Sven Milinski, Chris Nott, Rick Robinson, Jonathan Adams, and Paul Verschueren. *Patterns: Implementing an SOA Using an Enterprise Service Bus*. IBM, International Technical Support Organization, 2004.

[4] K Mani Chandy. Event-Driven Applications: Costs, Benefits and Design Approaches. *Gartner Application Integration and Web Services Summit*, 2006, 2006.

[5] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.

[6] Jean-Louis Maréchaux. Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus. *IBM Developer Works*, pages 1269–1275, 2006.

[7] Brenda M Michelson. Event-Driven Architecture Overview. *Patricia Seybold Group*, 2, 2006.

[8] Mike P Papazoglou and Willem-Jan Van Den Heuvel. Service Oriented Architectures: Approaches, Technologies and Research Issues. *The VLDB Journal*, 16(3):389–415, 2007.

[9] Doug Kaye. *Loosely Coupled: The Missing Pieces of Web Services*. RDS Strategies LLC, 2003.

[10] David A Chappell. *Enterprise Service Bus*. O'Reilly media, 2009.

[11] S. Ortiz. Getting on Board the Enterprise Service Bus. *Computer*, 40(4): 15–17, 2007. ISSN 0018-9162. doi: 10.1109/MC.2007.127.

[12] Saul Caganoff. The Benefits of an ESB, 2013. URL `http://www.soabloke.com/2008/01/31/the-benfits-of-an-esb/`. Accessed: 23/05/2013.

[13] Trinh Hoang Nguyen, Kamyar Rasta, Yohanes Baptista Dafferianto Trinugroho, and Andreas Prinz. Using Enterprise Service Bus for Offshore Wind Data Handling. In *Proceedings of 2012 9th IADIS International Conference on Applied Computing*, pages 83–90, 2012. ISBN 978-989-8533-14-2.

[14] Yohanes Baptista Dafferianto Trinugroho, Kamyar Rasta, Trinh Hoang Nguyen, Rune Werner Fensli, and Frank Reichert. A Real-Time Web-Based Health Monitoring System Based on Enterprise Service Bus. In *Proceedings of 2012 11th IADIS International Conference on WWW/Internet*, pages 165–172, 2012. ISBN 978-989-8533-14-2.

[15] Yohanes Baptista Dafferianto Trinugroho, Kamyar Rasta, Trinh Hoang Nguyen, Martin Gerdes, Rune Werner Fensli, and Frank Reichert. A Location-Independent Remote Health Monitoring System Utilising Enterprise Service Bus. *IADIS International Journal on WWW/Internet*, 10(2):88–106, 2012. ISSN 1645-7641.

[16] Kamyar Rasta, Trinh Hoang Nguyen, and Andreas Prinz. Intelligent Resource Management in Enterprise Service Bus. Technical report, University of Agder, 2013.

[17] Richard Y Wang. A Product Perspective n Total Data Quality Management. *Communications of the ACM*, 41(2):58–65, 1998.

[18] Diane M Strong, Yang W Lee, and Richard Y Wang. Data Duality in Context. *Communications of the ACM*, 40(5):103–110, 1997.

[19] Kuan-Tse Huang, Yang W Lee, and Richard Y Wang. *Quality Information and Knowledge*. Prentice Hall PTR, 1998.

[20] Mouzhi Ge and Markus Helfert. A Framework to Assess Decision Quality using Information Quality Dimensions. In *Proceedings of the 11th International Conference on Information QualityICIQ*, volume 6, pages 10–12, 2006.

[21] Barbara D Klein. Data Quality in The Practice of Consumer Product Management: Evidence from the Field. *Data Quality*, 4(1), 1998.

[22] Brian Snyder and Mark J Kaiser. Ecological and Economic Cost-Benefit Analysis of Offshore Wind Energy. *Renewable Energy*, 34(6):1567–1578, 2009.

[23] Russell L Ackoff. From Data to Wisdom. *Journal of Applied Systems Analysis*, 16:3–9, 2010.

[24] Hongjiang Xu. *Critical Success Factors for Accounting Information Systems Data Quality*. PhD thesis, University of Southern Queensland, 2009.

[25] Thomas C Redman. *Data Duality: Management and Technology*. Bantam Books, Inc., 1992.

[26] Sandra Geisler, Sven Weber, and Christoph Quix. Onotology-Based Data Quality Framewrok for Data Stream Applications. In *16th International Conference on Information Quality, November 2011, Adelaide, AUS*, 2011.

[27] Mouzhi Ge. *Information Quality Assessment and Effects on Inventory Decision-Making*. PhD thesis, Dublin City University, 2009.

[28] Matthew Bovee, Rajendra P Srivastava, and Brenda Mak. A Conceptual Framework and Belief-function Approach to Assessing Overall Information Quality. *International Journal of Intelligent Systems*, 18(1):51–74, 2003.

[29] Norbert Baumgartner, Wolfgang Gottesheim, Stefan Mitsch, Werner Retschitzegger, and Wieland Schwinger. Improving Situation Awareness

In Traffic Management. In *Proc. Intl. Conf. on Very Large Data Bases*, 2010.

[30] Richard Y Wang and Diane M Strong. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, pages 5–33, 1996.

[31] Leo L. Pipino, Yang W. Lee, and Richard Y. Wang. Data Quality Assessment. *Commun. ACM*, 45(4):211–218, April 2002. ISSN 0001-0782. doi: 10.1145/505248.506010.

[32] Yang W Lee, Diane M Strong, Beverly K Kahn, and Richard Y Wang. AIMQ: A Methodology For Information Quality Assessment. *Information and Management*, 40(2):133–146, 2002.

[33] William H DeLone and Ephraim R McLean. Information Systems Success: The Quest for the Dependent Variable. *Information Systems Research*, 3(1):60–95, 1992.

[34] Martin J Eppler. *Managing Information Quality: Increasing The Value of Information in Knowledge-intensive Products and Processes*. Springer, 2006.

[35] Richard Y Wang, Martin P Reddy, and Henry B Kon. Toward Quality Data: An Attribute-Based Approach. *Decision Support Systems*, 13(3): 349–372, 1995.

[36] Mica R Endsley and WM Jones. Situation Awareness. *The Oxford Handbook of Cognitive Engineering˜ autofilled˜*, page 88, 2013.

[37] Lukasz Golab and M Tamer Özsu. Issues in Data Stream Management. *ACM Sigmod Record*, 32(2):5–14, 2003.

[38] Gulnoza Ziyaeva, Eunmi Choi, and Dugki Min. Content-Based Intelligent Routing and Message Processing in Enterprise Service Bus. In *Convergence and Hybrid Information Technology, 2008. ICHIT'08. International Conference on*, pages 245–249. IEEE, 2008.

[39] Xiaoying Bai, Jihui Xie, Bin Chen, and Sinan Xiao. Dresr: Dynamic Routing in Enterprise Service Bus. In *e-Business Engineering, 2007.*

ICEBE 2007. IEEE International Conference on, pages 528–531. IEEE, 2007.

[40] Dimka Karastoyanova, Branimir Wetzstein, Tammo van Lessen, Daniel Wutke, Joerg Nitzsche, and Frank Leymann. Semantic Service Bus: Architecture and Implementation of A Next Generation Middleware. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 347–354. IEEE, 2007.

[41] Robert Battle and Edward Benson. Bridging the Semantic Web and Web 2.0 with Representational State Transfer (REST). *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):61–69, 2008.

[42] AW Manyonge, RM Ochieng, FN Onyango, and JM Shichikha. Mathematical Modelling of Wind Turbine in a Wind Energy Conversion System: Power Coefficient Analysis. *Applied Mathematical Sciences*, 6(91): 4527–4536, 2012.

[43] Jose de Jesus Rubio, Maricela Figueroa, Jaime Pacheco, and Manuel Jimenez-Lizarraga. Observer Design Based in the Mathematical Model of a Wind Turbine. *Int. J. Innovative Comput. I*, 7(12):6711–6725, 2012.

[44] International Organization for Standardization. *ISO 5725-2: 1994: Accuracy (Trueness and Precision) of Measurement Methods and Results-Part 2: Methods for the Determination of Repeatability and Reproductibility*. International Organization for Standardization, 1994.

[45] Paul D Sheriff. *Fundamentals of N-Tier Architecture*. PDSA, Inc., 2006.

[46] Bruce E Bargmeyer et al. Metadata Standards and Metadata Registries: an overview. *Citeseer*, 2000.

[47] Wendy L. Martinez. Graphical User Interfaces. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(2):119–133, 2011. ISSN 1939-0068. doi: 10.1002/wics.150. URL `http://dx.doi.org/10.1002/wics.150`.

[48] Chris Ullman and Lucinda Dykes. *Beginning Ajax*. Wrox, 2007.

[49] Vladislav Akhmatov and Hans Knudsen. An Aggregate Model of a Grid-Connected, Large-Scale, Offshore Wind Farm for Power Stability InvestigationsImportance of Windmill Mechanical System. *International Journal of Electrical Power & Energy Systems*, 24(9):709–717, 2002.

[50] Mark Desholm. Thermal Animal Detection System (TADS): Development of a Method for Estimating Collision Frequency of Migrating Birds at Offshore Wind Turbines. Technical report, National Environmental Research Institute, 2003.

[51] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The Many Faces of Publish/Subscribe. *ACM Comput. Surv.*, 35(2):114–131, June 2003. ISSN 0360-0300. doi: 10.1145/857076.857078. URL `http://doi.acm.org/10.1145/857076.857078`.

[52] Thomas R Gruber et al. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[53] F. AlShahwan and K. Moessner. Providing SOAP Web Services and RESTful Web Services from Mobile Hosts. In *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*, pages 174–179, 2010. doi: 10.1109/ICIW.2010.33.

[54] Gavin Mulligan and Denis Gracanin. A comparison of soap and rest implementations of a service based interaction independence middleware framework. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 1423–1432. IEEE, 2009.

Appendix A

Appendix A. An Attached Publication

Title: A Framework for Data Quality Handling in Enterprise Service Bus.

Affiliation: Department of Information and Communication Technology, University of Agder, Norway.

Submission status: Submitted to International Conference on Innovative Computing Technology (INTECH 2013), London, August 29-31, 2013

A Framework for Data Quality Handling in Enterprise Service Bus

Kamyar Rasta*, Trinh Hoang Nguyen**, Andreas Prinz**
Department of Information and Communication Technology
University of Agder
N-4898 Grimstad, Norway
*kamyar11@student.uia.no
**{trinh.h.nguyen, andreas.prinz}@uia.no

***Abstract* - Enterprise Service Bus (ESB) is proposed to address the application integration problem by facilitating communication among different systems in a loosely coupled, standard-based, and protocol independent manner. Data sources are maintained out of the ESB's control and there should be a mechanism to select the most suitable data source among all available data sources. Especially, when two or more data sources about the same object. For instance, it is normal to use more than one sensor to measure pressure or temperature at a particular point. Data quality can play an important role in selecting data sources in ESB since quality of data is an essential factor in the success of organizations. There is no built-in component in the current ESB platforms to handle data quality. In this paper, we present a flexible and comprehensive data quality framework for managing resources in ESB. The framework is independent of ESB platforms. We evaluate our framework using four different scenarios within the wind energy domain.**

I. INTRODUCTION

The growing number of applications distributed across the Internet, interacting with a large number of users has led to a need for a better communication platform between users and applications. Service-Oriented Architecture (SOA) increasingly gains momentum in industry as a mean to facilitate communication and collaboration between different systems running on multiple platforms. SOA is based on loosely coupled, distributed, and interoperable services [1].

Enterprise Service Bus (ESB) has been regarded as a promising technology to build an infrastructure for SOA [1],[2]. ESB can be used as a centralized solution to solve the data integration problem effectively in various domains such as power system, eHealth, and oil & gas. Although ESB handles communication between applications, it does not support a way to select the most suitable data source among all possible options. A widespread issue in data integration is the management of data sources with insufficient data quality. For example in offshore wind energy, a couple of sensors are deployed on a windmill and they frequently measure and deliver the data to the users and applications by means of services. As sensors are prone to failures their results might be inaccurate, incomplete, and inconsistent [3]. Therefore, there should be a way for users and applications to specify the desired quality level of the data sources. Only when the data source has the requested quality descriptions it would be used for further processing.

In enterprise applications with a potentially large number of data sources with quality information, handling the data quality at user level is a challenging problem. To tackle this problem, the data quality process should be moved from the user level to the middleware level (i.e. ESB). However, there is no data quality component in the current ESB

platforms. Therefore, knowledge about specifications of such a component is desirable to ensure the quality of data sources in ESB.

In this paper, we present a flexible and comprehensive data quality framework for managing data sources in ESB. We propose a way to describe, measure, and compare the quality of data sources based on the existing data quality frameworks. In addition, an approach is presented for selecting a data source with the most proper data quality according to given quality criteria. We provide dynamic binding and invocation of resources in ESB to make implementation details transparent to users and applications. This work also deals with specifying a way to handle quality of data sources that are derived from the other data sources. The evaluation of our approach in four scenarios in wind energy domain has shown that our framework provides the required flexibility, extensibility, and performance.

The rest of the paper is organized as follows: Section II describes the core terms used in the paper and data quality dimensions. Section III proposes a framework for quality-based resource management in ESB. Section IV describes a prototype implementation of the proposed framework in terms of four scenarios within the wind energy. Section V presents some related work. Finally, section VI highlights the primary contributions of this paper and the conclusion reached.

II. DATA QUALITY AND SOURCES

This section describes the important concepts in this paper such as data source and data quality. Afterwards, different dimensions of data quality are presented.

A. Data sources

A data source consists of two parts: data item and quality description. These are described as follows: A data source consists of three parts: data item, data item description, and quality description. These are described as follows:

- *Data item:* this is a sensor data. In particular, it is a stream of signals coming from a source. We call each signal in the stream a data point, for instance, 6 for wind speed at a specific date and time. The whole stream is called a data item, for example, 4000 wind speed values from the 1^{st} of May to the 1^{st} of June. The data item is provided by a service deployed on a data provider. The services can expose available sensor data to the users in a standard manner.
- *Data source description:* is the information about the data source to describe the data source in the context of the domain. The data item is only a sequence of numbers and there is no information what kind of data, unit, and protocol are used to represent this data item. While the data source description provides that information and for instance, it specifies the data type is wind speed, the unit is m/s, and the protocol is REST [4]. In our system data source description comes from SOA components. In particular, this description is in the service registry and there is an address to get access to it which is stored in a specific repository.
- *Quality description:* is a description of the quality of data source. For example, the quality description of a data source determines that the completeness is 80% and timeliness is 1 ms. Quality description is conceptually part of data source description but because of implementation simplicity we separate it from data source description.

In this work, we discuss two types of data sources. The first one is real data source, which denotes a regular data source connected to a real sensor. The second one is virtual data source, which is computed by combining one or more real data sources. There is an inevitable assumption for data source combinations: the data source description of the combining real data sources should be almost equivalent. Obviously, it is impossible to combine a wind speed data source with a temperature data source. However, there are some cases where the combination becomes possible using a converter even though the real data sources are not completely equivalent. For example, if there are two wind speed data sources with different measurement units (e.g. m/s and km/h), it is possible to convert the units using a converter function before combining the two data sources.

Since the virtual data sources are generated by combining the real data sources, they also consist of three parts: data item, data source description, and quality description. The virtual data item is combined by a formula, e.g. average of data items from the real data sources. The virtual data source description is often remained the same to the real data sources or at least to one of the data sources. The quality description of the virtual data source is computed by combining the quality descriptions of the real data sources.

B. Data quality dimensions

Many data quality dimensions have been proposed by various research projects. Table A.1 shows an overview of data quality dimensions used in eight studies. The most commonly employed quality dimensions are *accuracy*, *completeness*, and *timeliness*.

We classify the quality dimensions into two groups: measurable and user interface. Measurable contains a set of dimensions that can be computed and used for data collection. On the other hand, the user interface group contains data presented to users in terms of diagrams or other tools. Typically, user interface is used to support data presentation.

Since sensors will be the only data sources, the measurable dimension group is chosen to be discussed in this work. So we pick up quality dimensions that are not only measurable, but also applicable in the wind energy domain. Another reason for this selection is that since we do not have data processing, the presentation of data is out of scope of this work. The chosen dimensions are defined as follows:

Accuracy: we define accuracy as how close the observed data are to reality. Accuracy has two parts: precision and trueness.

- *Precision* is the closeness of agreement in individual results which is the standard deviation.
- *Trueness* is defined as the mean value of the difference of data source to the reality. We assume that the sensors are calibrated. Since we handle calibrated sensors, the trueness is very close to the zero. Then we only use precision as the accuracy in our system.

Assume N is the total number of data points in data source D. The d_i is a data point of D and r_i is a reality value. The x_i is the difference between the data point and the reality. The μ denote the trueness. The accuracy of data source D is calculated Eq. (A.1):

TABLE A.1: Data quality dimensionts

	Classifications									
Dimensions	Delone & Mclean 1992 [5]	Wang & Strong 1996 [6]	Pipino et al. 2002 [7]	Lee et al. 2002 [8]	Bovee et al. 2003 [9]	Eppler 2006 [10]	Baum gartner et al. 2010 [11]	Geisler et al. 2011 [12]	Number of studies	**Measurable & User interface**
Accuracy	X	X	X	X	X	X	X	X	8	**M**
Completeness	X	X	X	X	X	X	X	X	8	**M**
Timeliness	X	X	X	X	X	X	X	X	8	**M**
Consistency		X	X	X	X	X	X	X	7	**M**
Access security		X	X	X		X			4	**M**
Data volume		X	X	X				X	4	**M**
Relevancy	X	X	X	X					4	**U**
Accessibility		X	X	X	X	X			5	**U**
Confidence							X	X	2	**U**
Coverage							X		1	**U**
Objectivity		X	X	X					3	**U**
Believability		X	X	X					3	**U**
Reputation		X	X	X					3	**U**
Value-added		X	X						2	**U**
Interpretability		X	X	X	X	X			5	**U**
Understandability	X	X	X	X					4	**U**
Conciseness	X	X	X	X		X			5	**U**

$$\begin{aligned} &Accuracy(D) = \tfrac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2, \text{ Where} \\ &\mu = 0 \text{ and } x_i = d_i - r_i \text{ then} \\ &Accuracy(D) = \tfrac{1}{N}\sum_{i=1}^{N}(d_i-r_i)^2 \end{aligned} \tag{A.1}$$

The unit of accuracy is largely depends on the unit of the data source.

Completeness: is defined as the ratio of correctly received data points to the total number of sent data points. Let D denotes a data source and R is the reference data source (reality). If the total number of sent data points in a given interval is N_R and N_D is the number of data points in D, then the completeness data source D can be calculated by Eq. (A.2):

$$Completeness(D) = \frac{N_D}{N_R} \tag{A.2}$$

For example when there is a graph of wind speed data with 4000 data points and only 3200 of them are received the completeness of wind speed graph becomes 80%.

Additionally, the distribution of incompleteness could be varied. For example there is a difference between missing a data point every second and missing of all data points in

one hour in the middle of the transmission.

Timeliness: is the average time difference between the producing of the data and the receiving of the data. Assume the t(r_i) denotes the time when the reference data point i is produced and t(d_i) denotes the time when the data point d_i is received. The timeliness of data source D is calculated Eq. (A.3):

$$Timeliness(D) = \frac{\sum_{i=0}^{N} t(d_i) - t(r_i)}{N} \tag{A.3}$$

Consistency: The extent to which the domain constraints have been met. Assume N_C is the number of consistent data points in data source D and N_R is the total number of data points in reference data source, then *Consistency(D)* represents the consistency of D and it is computed by Eq. (A.4):

$$Consistency(D) = \frac{N_C}{N_R} \tag{A.4}$$

For example, we define $S = \{x \in \mathbb{R} \mid 0 \leq x \leq 150\}$ is the set of valid data points for a wind speed data source with 4000 data points. So if we receive 3200 data points with values between zero and 150, the consistency of the D is 80%.

III. A DATA QUALITY-BASED FRAMEWORK

This section describes our proposed framework for quality-based resource management. *A. Design requirements* In order to develop scalable data quality-based applications, the proposed framework has to be general-purpose and should not adhere to specific dependencies of any scenario. To achieve this requirement the design should allow for scalability with respect to the framework components, quality dimensions, and data source combination methodologies used. Data quality dimensions have an important role in this approach and imply some design requirements.

- The framework must allow data providers to define what quality dimensions could be taken into account.The user needs to acquire quality criteria from different data sources in a disciplined way.
- Another important requirement is the independence of the framework from any ESB platform. The framework should attach and work with ESB regardless of the platform type. However, in practice, some minor modifications are acceptable.
- The framework should be independent of the target domain. Nevertheless, to be more realistic, a few changes are acceptable in different application domains.

B. The proposed framework Taking the predefined requirements into consideration we design a framework as shown in Fig. A.1. The framework architecture uses a multilayer approach to handle data quality in ESB based on the previously set out requirements. The framework consists of three layers: *the presentation layer*, *the quality-based resource management layer*, and *the data provider layer*.

The *Presentation layer* formalizes the submitted request from the user and passes the request to the second layer. The data source request is a formal description of the user's submitted request. The *Quality-Based Resource Management layer* is employed to select

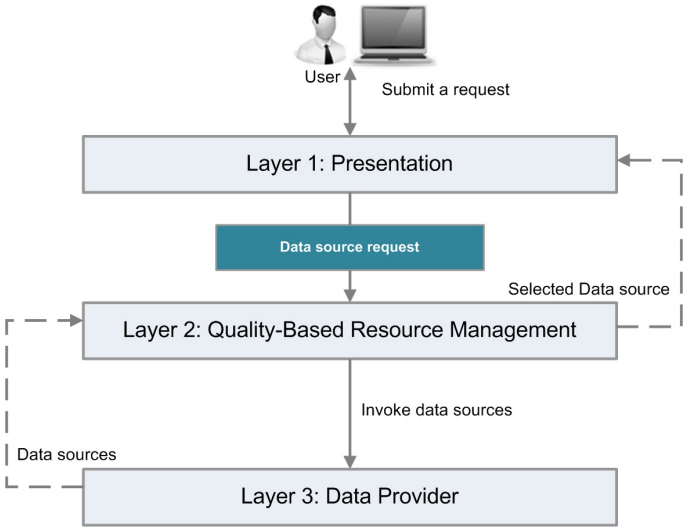

FIGURE A.1: An overview of the framework architecture.

the best data source with respect to the requested data, quality criteria, and constraints. This layer finds out the most appropriate data source for the user's request and gives that data source to the user. The *Data provider layer* involves a set of external data providers including the data, quality descriptions, and the way of access to the data.

The proposed multilayer design, separating the functions gives several benefits. This design allows the framework to employ different component technologies, ESB platforms, and deployment environments. It also helps the isolation of data sources and their potential failures that are passed to the proper layers to be dealt with.

C. Presentation layer Fig. A.2 shows the structure of the *Presentation layer*. The *Data source request* is an entry point to the data quality handling process. It defines users' requests precisely and is generated in three steps.

- Step 1: A user starts with retrieving the domain information and quality dimensions from a *Domain repository*, for example, the user receives a list of available wind energy data together with a list of quality dimensions
- Step 2: The user selects the require data name, quality dimensions, and constraints, for example, the user selects wind speed with completeness of more than 80% from 1^{st} of May to the 1^{st} of June.
- Step 3: The selected data name, quality dimensions, and constraints put together and the *Data source request* is generated.

B. Quality-Based Resource Management Layer The data quality handling process is split into three sub-layers *ESB*, *Service Manager*, and *Invocation Manager* as shown in Fig. A.3. The ESB contains one module that is called *Request Manager* to connect users to ESB and other part of the system. The *Request Manager* forwards the *Data source request* to the *Service Manager*. This sub-layer is in charge of finding proper data sources for user and it contains two core components: the *Quality Manager* and the *Service-Quality Database*.

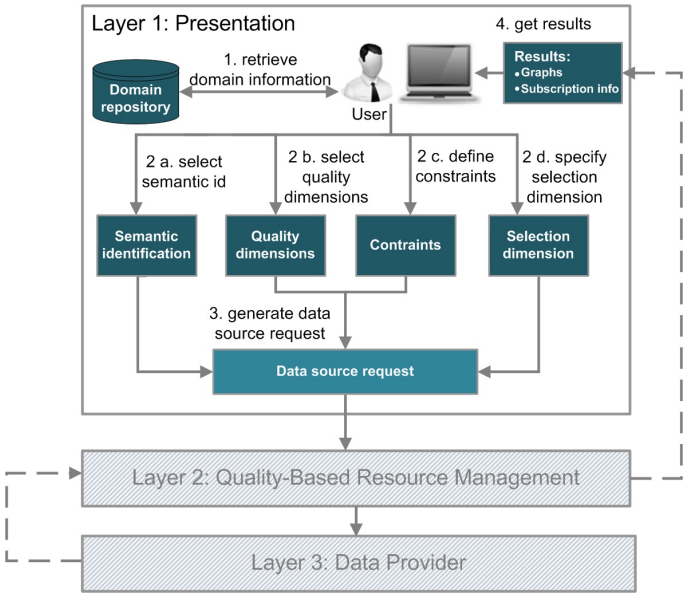

FIGURE A.2: The presentation layer.

The *Quality Manager* selects the most appropriate data sources in terms of acquired quality aspects and constraints from the *Service-Quality Database* where all information about data sources, quality dimensions, and their corresponding values are stored. It generates the *Selected data source list* consists of the necessary information of the suitable data sources such as the data source address, parameters, and values. The *Service Manager* passes the list to the next sub-layer which is the *Invocation Manager*. As we mentioned the data sources are provided through a number of services and then it is essential to bind and invoke the data sources in dynamically. The *Dynamic Invoker* reads the data sources in the *Selected data sources list* and invokes them.

The idea of three-layered approach has been introduced for some reasons. Firstly, splitting the tasks of data quality handling process into separate layers enhances the extendibility and scalability of the framework. Secondly, having a separate ESB layer isolates this platform from other parts of the framework. Consequently, the independence of the framework from ESB platforms is guaranteed and the ESB platform replacement becomes a simple process. As a result, a wider number of users can take advantages of our framework regardless of their ESB technology.

E. The Data Provider layer Fig. A.4 illustrates the *Data provider layer* and represents a set of external data sources.

As aforementioned, a data source has two parts a data item and a quality description. The data item provides the data in the terms of services with local databases of sensors data. These services are accessible through their service descriptions. The *Dynamic Invoker* binds and invokes the data item of the selected data sources in its list (Step 1 Fig. A.4). When the service is invoked, it executes a query on its local *Data source database* and retrieves the stored data points in database (Step 2).

IV. IMPLEMENTATION

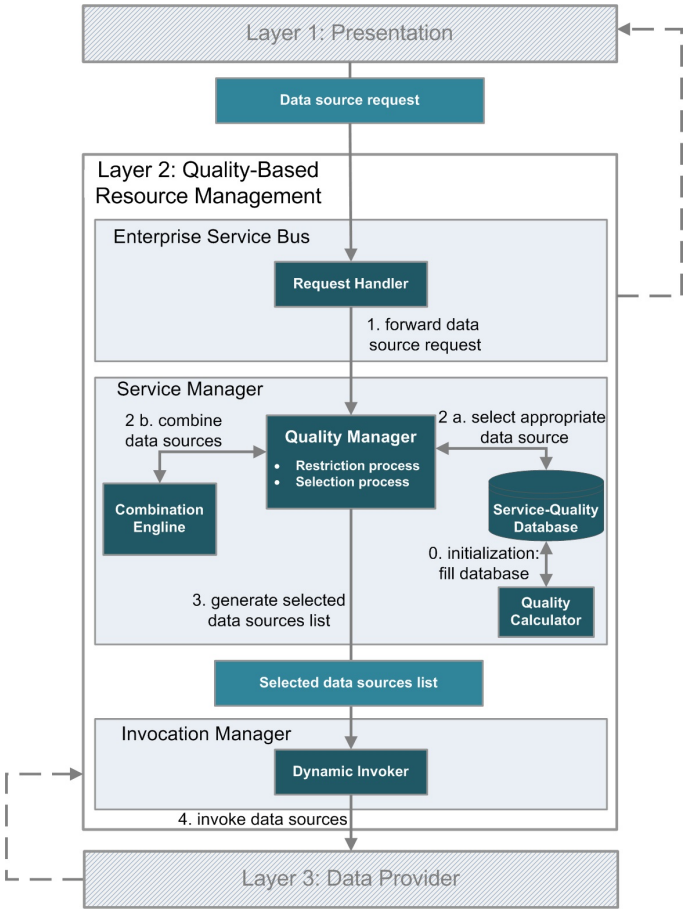

FIGURE A.3: The quality-based resource management layer.

This section present an implementation of the framework. A prototype system has been developed. The prototype contains three main parts, web-based client application, integration server, and data provider services. Fig. A.5 shows these components and their interactions.

Users use the web-based client application to send requests to the integration server for data provided by data providers. These three high level components are fitted to the framework layers that have been shown in Fig. A.1.

A. The prototype description

We use different wind data source services as third party data providers. SOAP-based (Simple Object Access Protocol) protocols are used to handle the communication between the integration server and data provider services. In order to handle the messages coming from the integration server to the client application, AJAX (Asynchronous JavaScript and XML) technologies are used. The integration server consists of following four main components:

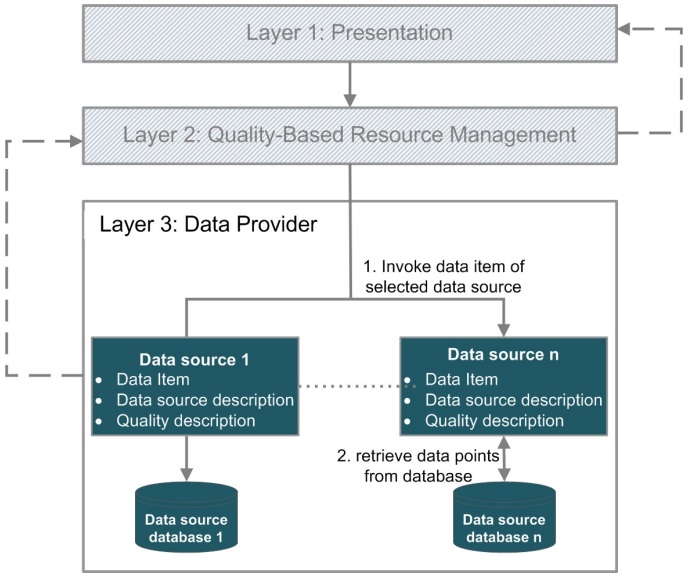

FIGURE A.4: The data provider layer.

- MuleESB[1]: is an open source enterprise service bus framework and we use it as the communication backbone of the system because of its simplicity, large number of connectors, transports, and its light weight foot prints [38].
- Service Manager: is a service and deployed on Internet Information Services (IIS) web server. It contains a Quality Manager module responsible for analysing the data quality. The Quality Manager finds the best data source information and asks the Invocation Manager to invoke the required data.
- Invocation Manager: is also a service that dynamically binds and invokes the selected data sources.
- Service-Quality database: is a SQL Server[2] relational database that stores information about all entities of the system in two main categories: service and quality information. The first category is used to store information of services and invoke them. The second category stores information required for data quality handling. Quality Manager executes query on quality information category for selecting the required data item and the result of query typically is a list of Web Service Definition Language (WSDL) address, required operations, and their input/output parameters. These lists are delivered to the Dynamic Invoker Service for subsequent biding and invocation.

Data provider services are third party services that provide data and quality information of the data. We use Java API for XML Web Services (JAX-WS) client to send data and quality using SOAP messages. A data provider announces the quality descriptions by publishing the quality descriptions as an extra service. This service is characterized by

[1] http://www.mulesoft.org
[2] http://www.microsoft.com/en-us/sqlserver/default.aspx

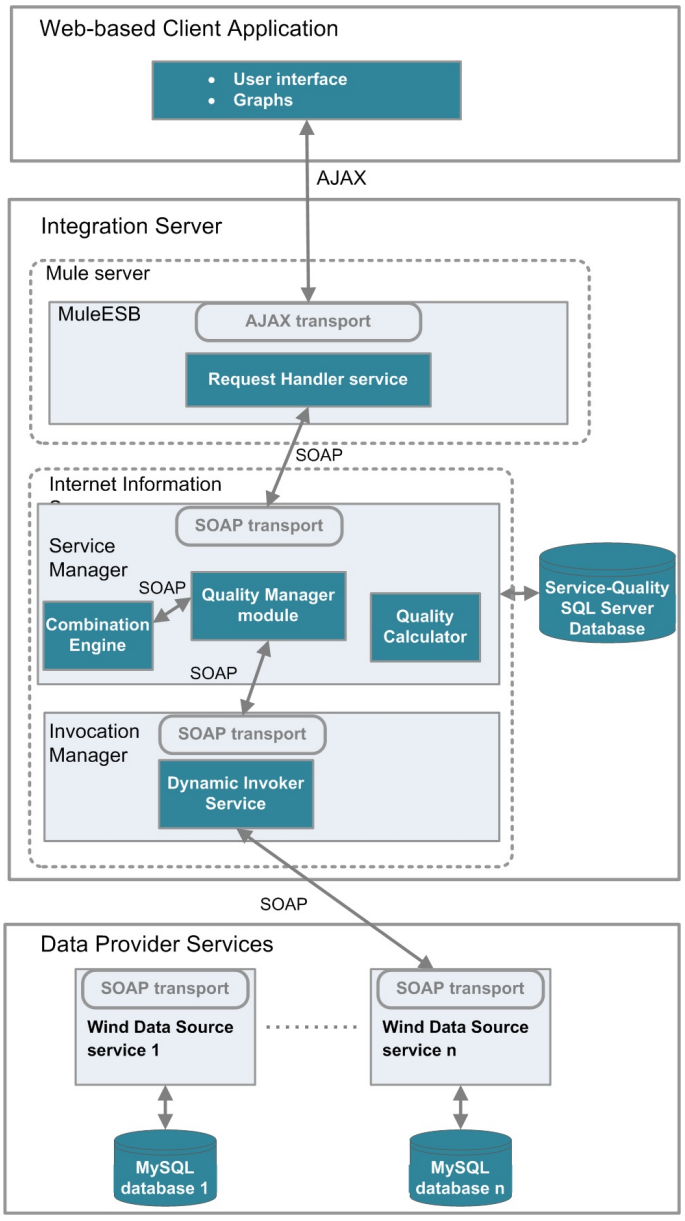

FIGURE A.5: A prototype implementation of the framework.

a suffix of "_Quality". For example, Get_WindSpeed service has a corresponding quality description service that is called Get_WindSpeed_Quality.

B. Scenarios To illustrate how the data quality resource management can be done using the proposed architecture, we implement the following scenarios in the scope of the offshore wind domain. A windmill is one of the main components of the offshore wind management systems and a number of wind sensors (e.g., wind speed sensor) are attached to it. As offshore windmills are located far from the coastline where the weather is harsh, sensors are subject to the moisture and corrosion. Therefore, the quality of the data produced by them can be negatively influenced [3].

A data quality-based resource management system in offshore wind energy domain requires measurements of different wind information from wind sensors. Each sensor can send the measurements to the integration server through a third party wind service, in which performs a specific data processing and relays the information. A user can send a request to the sensors data via the integration server. The integration server process the request and response to the user.

Scenario 1

Scenario description: Given all the data sources are real. There is only one quality criterion (i.e. completeness) for each data source. The quality description is available for each data source. Fig. A.6 shows the scenario 1 where the data is provided by wind sensors through a number of wind web services. A single data source consists of two elements; a data item (i.e. wind service) and a data quality description which is a set of data quality dimensions (i.e. completeness in this case). There are three data sources for wind speed with different completeness values which is given by the sensor provider in device specification in terms of a percentage.

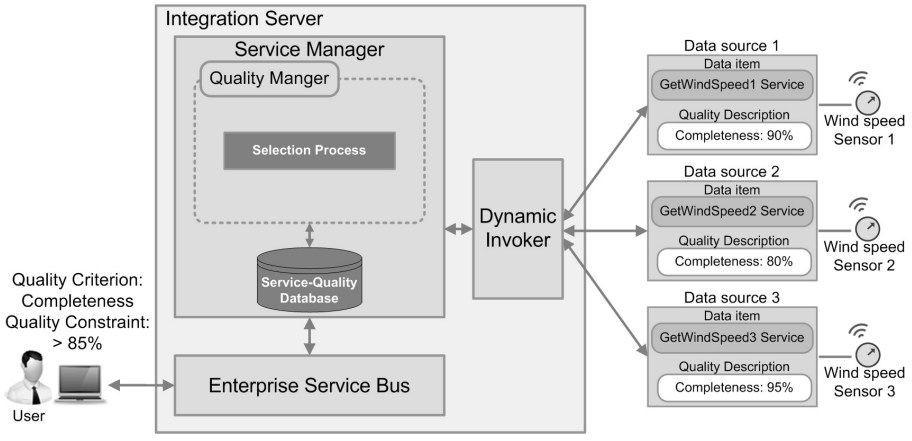

FIGURE A.6: Scenario 1.

A user issues a request for wind speed with at least completeness of 85%. The *Service Manger* receives this request and gives it to the *Quality Manger*. The *Quality Manger* retrieves the quality descriptions of existing data sources from its quality database and finds the data sources with appropriate completeness. If there are more than one data sources that meet the user specified requirements, the Quality Manager choose the data source with the highest completeness. In Fig. A.6 there are two appropriate data sources;

data source 1 and 3 because they have the completeness of 90% and 95% respectively. The Quality Manager selects the data source 3. Figure A.7 shows the graph of data source 3.

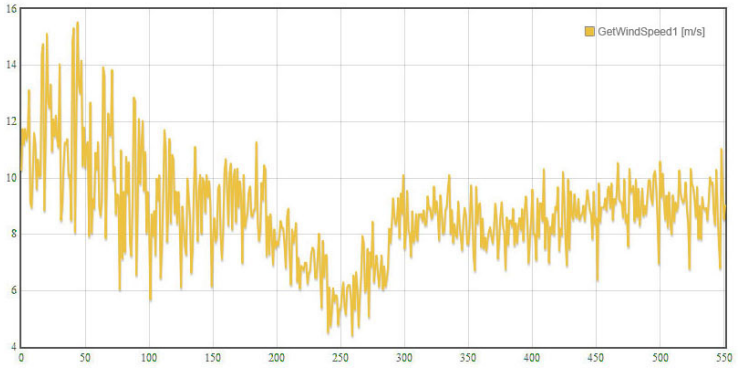

FIGURE A.7: The graph of Data source 3 that is selected by *Quality Manager* and is shown for the user.

Scenario 2

Scenario description: Fig. A.8 shows the second scenario where there are one virtual (derived) data and two real data sources in the system. There is only one criterion (i.e., completeness) for each data source. The quality descriptions are available for all data sources.

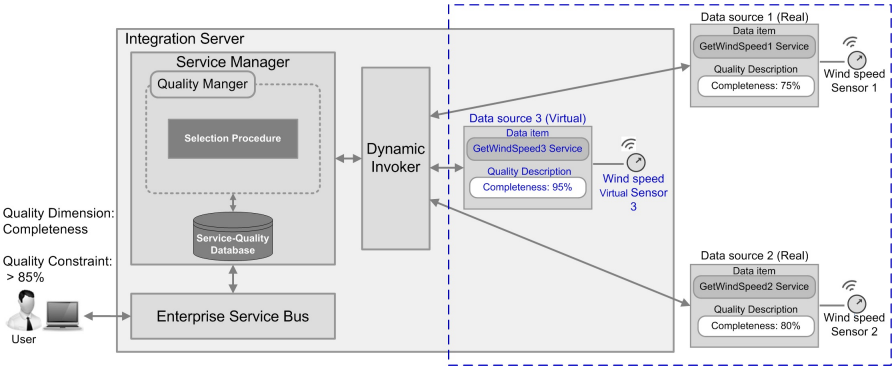

FIGURE A.8: Scenario 2.

The virtual data source is incorporated by a composed service (out of other services) and data quality descriptions. There are different operations to calculate both combined data and its quality descriptions, e.g., average of the two data sources and "first in first pick up" operations. We assume the later operation in this scenario which means for each data point in the data item, the system first pick up the data point from the data source 1 if available, otherwise pick up from data source 2.

The user issues a request for wind speed with completeness of at least 85%. The completeness of data sources 1, 2, and 3 are 75%, 80%, and 95% respectively which means by combining two real data sources a virtual data source with higher completeness

is generated. The reason is that each of the data sources 1 and 2 has missed some parts of the data but when they are composed, each data source might fills the missing part of the other one. Therefore, Quality Manager chooses the virtual data source.

Scenario 3

Scenario description: Given all the data sources are real. There are multiple quality criteria for data sources. The quality descriptions are available. Fig. A.9 shows the third scenario where there are four real wind services each with four quality dimensions of accuracy, completeness, timeliness, and consistency.

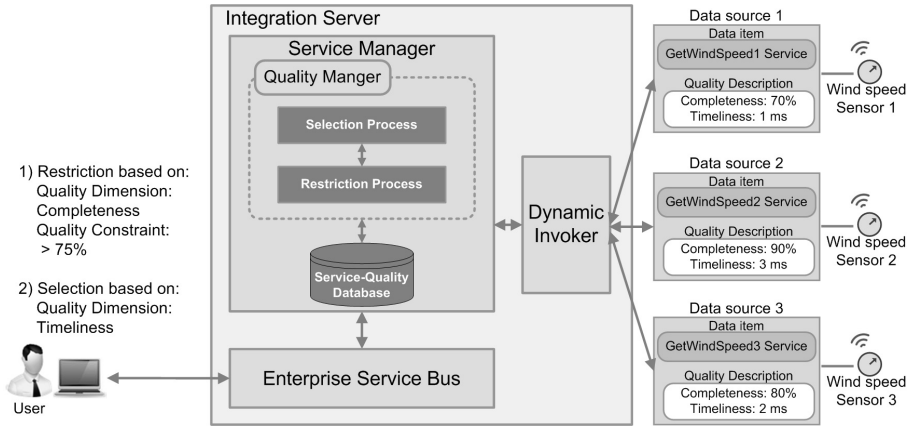

FIGURE A.9: Scenario 3.

The Quality Manager uses two processes to find the best data source for the user; restriction process and selection process. The purpose of the restriction process is to detect and filter out outliers from the results according to the given quality dimensions and constraints. The restriction process prepares a list of available good data sources and hands in this list to the selection process. The basic idea of selection is to find the best data source among good data sources according to selection quality dimension.

A user sends a request for wind speed with two class of quality dimensions; restriction quality dimensions and selection quality dimensions. The *Service Manger* receives this request from the user and gives it to the *Quality Manger*. At restriction process the Quality Manage filters the data sources based on user defined restriction quality dimensions (i.e. Completeness $> 70\%$, Timeliness < 4 ms, and Consistency $>= 80\%$). Only two data sources 3 and 4 meet the restriction process requirements, and are transferred to the selection process. Since the user chooses "Accuracy" as his selection quality dimension, the most accurate data source between data sources 1 and 2 should be chosen, which is data source 4 (with accuracy of 85%).

Publish/Subscribe Scenario

Scenario description: In the publish/subscribe pattern, there are some publishers and subscribers in the system. Subscribers ask for subscription to the publishers and whenever a publisher publishes any data item, all of its subscribers would receive the published data items. In this scenario when a user sends a request to the integration server, Quality Manger finds the best data source to be subscribed, and then Service Manger gives the subscription information to the user. This information is an ESB channel name. When user receives the channel name, it subscribes to the channel by calling a function in

the MuleESB called Mule.Subscribe ("channelname"). Once the user subscribes to the channel he will receive all data that are pushed by the data providers. Basically in publish/subscribe pattern scenario there is no dynamic invoker component because the data sources are no more invoked from integration server. Fig. A.10 shows data providers that are pushing the data and there are three channels for each them.

FIGURE A.10: Publish / subscribe scenario, data providers push data.

Fig. A.11 illustrates once the user is subscribed to a data source, he receives real-time data of that data source.

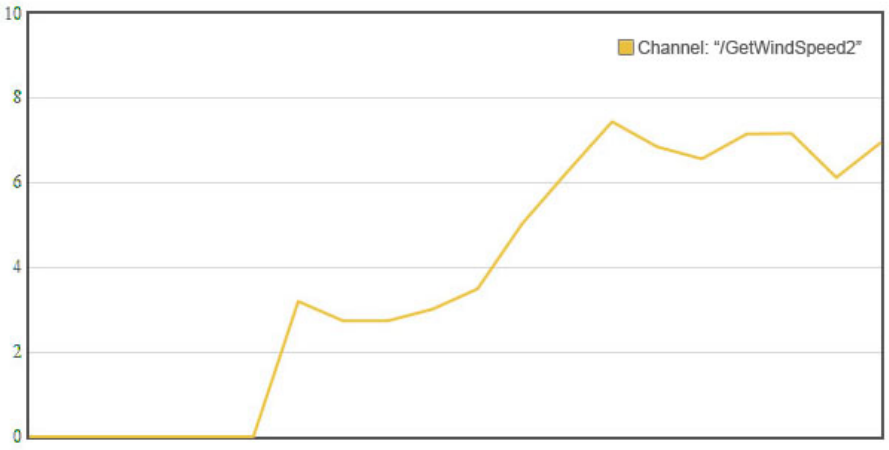

FIGURE A.11: Publish / subscribe scenario, user is subscribed to one data source and receives real-time data.

V. RELATED WORK AND DISCUSSION

There are some previous works related to resource selection in ESB that are interesting to consider. Authors in [13] have proposed a multi-layered framework to support the content-based for intelligent routing path construction and message routing. Their approach facilitates the data source selection based on the message content. Dynamic service selection and composition in ESB have presented by [14]. They use abstract service names to define an Abstract Routing Path (ARP) and replace this path with uniform resource identifier (URI) of the real data provider at run-time. Their model makes the path selection in ESB transparent and service composition configuration dynamic. Authors in [15] have introduced a middleware representing a combination of SOA and Semantic Web and they call it Semantic Service Bus. It enhances the processes of service discovery, routing, composition, etc., by employing semantic description of services.

The common fact in all aforementioned works is that the resource selection process is based on data itself. Different from these approaches, our approach is based on quality description of the data.

VI. CONCLUSION

As ESB is gaining more attention from industry, there is an increasing need to specify ESB's resource management in a disciplined manner. The proposed framework presents a novel data quality-based resource management model for ESB. We propose a way for data quality handling in a structured and scalable way.

Users of ESB require measured data and quality descriptions coming from different sensors to be sent directly. Therefore, the users have to go deep to the system and know lots of the details of the data providers and their quality specifications. The framework moves the data quality process from the user level to the integration server (middleware level). It improves the overall performance of the system and lifts the user's experience in the sense of decreasing the details that user has to deals with in order to get access to data. This framework makes an abstraction that is a data model and the user is just relating to the domain description and he starts with the domain description and asks a data source with specific quality aspects and constraints, and then the framework selects the most suitable data provider for the user.

The potential directions for future work include more sophisticated methods for combining the data sources, supporting the data quality handling when the quality description is not available for some data sources. In later, there should be a method to find out the quality out of different received data. Another next step is the use of ontologies and semantic technologies in the selection process to improve the probability of finding suitable data source for users.

REFERENCES

[1] D. K. Barry, *Web services and service-oriented architectures: the savvy manager's guide.* Morgan Kaufmann, 2003.
[2] T. Erl, *Service-oriented Architecture: Concepts, Technology, and Design.* Pearson Education India, 2006.
[3] B. Snyder and M. J. Kaiser, "Ecological and economic cost-benefit analysis of offshore wind energy," *Renewable Energy*, vol. 34, no. 6, pp. 15671578, 2009.
[4] R. Battle and E. Benson, "Bridging the Semantic Web and Web 2.0 with Representational State Transfer (REST)," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, no. 1, pp. 6169, 2008.
[5] W. H. DeLone and E. R. McLean, "Information systems success: the quest for the dependent variable," *Information systems research*, vol. 3, no. 1, pp. 6095, 1992.
[6] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *Journal of management information systems*, pp. 533, 1996.
[7] L. L. Pipino, Y. W. Lee, and R. Y. Wang, "Data quality assessment," *Communications of the ACM*, vol. 45, no. 4, pp. 211218, 2002.
[8] Y. W. Lee, D. M. Strong, B. K. Kahn, and R. Y. Wang, "Aimq: a methodology for information quality assessment," *Information and management*, vol. 40, no. 2, pp. 133146, 2002.
[9] M. Bovee, R. P. Srivastava, and B. Mak, "A conceptual framework and belief-function approach to assessing overall information quality," *International journal of intelligent systems*, vol. 18, no. 1, pp. 5174, 2003.
[10] M. J. Eppler, *Managing information quality: increasing the value of information in knowledge-intensive products and processes.* Springer, 2006.
[11] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, and W. Schwinger, "Improving situation awareness in traffic management," in *Proc. Intl. Conf. on Very Large Data Bases,* 2010.
[12] S. Geisler, S. Weber, and C. Quix, "Ontology-based data quality framework for data stream applications."
[13] G. Ziyaeva, E. Choi, and D. Min, "Content-based intelligent routing and message processing in enterprise service bus," in *Convergence and Hybrid Information Technology, 2008. ICHIT'08. International Conference on.* IEEE, 2008, pp. 245249.
[14] X. Bai, J. Xie, B. Chen, and S. Xiao, "Dresr: Dynamic routing in enterprise service bus," *in e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on.* IEEE, 2007, pp. 528531.
[15] D. Karastoyanova, B. Wetzstein, T. van Lessen, D. Wutke, J. Nitzsche, and F. Leymann, "Semantic service bus: architecture and implementation of a next generation middleware," in *Data Engineering Workshop, 2007 IEEE 23rd International Conference on.* IEEE, 2007, pp. 347354