Prototype for a IEC 61400-25 Compliant Generic Server

Supervisors:
Bjarne Poulsen, DTU-IMM
Knud Ole Helge Pedersen, DTU-CET

Students:
Andreas Kargård Olsen - s022104
Baris Özdil - s041945

Kongens Lyngby 2006
IMM-MSc-2006

Technical University of Denmark
Informatics and Mathematical Modeling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk
Thesis number 96

Summary

IEC61850 has defined a family of standards for the power grid. E.g. the new IEC 61400-25 defines protocols for communication, control, and monitoring of wind power plants (WPP). This standard includes a wide range of mandatory and optional elements, ranging from security, communications interface, and system speed. This enables control and monitoring to be handle communication in a standardized and secure way. An analysis focusing on isolating the necessary requirements of IEC61400-25 has been carried out to create a generic prototype which can be used by WPP vendors where main communications interface of the prototype utilizes web services. The prototype is comprised of several independent modules to allow the possibility of choosing a fully customizable setup by the end user. Configuration of the system need to be done in an easy way, ensuring a flexible and reusable system, where different choices for the system can be added or left out depending on user specifications. From the requirements a prototype with the purpose of examining the key aspects of these definitions has been elaborated.

Resume

IEC61850 har defineret en familie af standarder til brug i el systemet. IEC 61400-25 definer de data modeller og protokoller som kan benyttes til at kommunikere, kontrollere og overvåge en vind mølle. (Wind Power Plant - WPP). Denne standart består af en lang række obligatoriske og frivillige elementer, som strækker sig over områder så som sikkerhed, kommunikations interface, og system hastighed. Disse sørger for at kontrol og overvågning kan blive udført på en standardiseret og sikker måde. En analyse er foretaget, med henblik på at afdækkende de nødvendige krav som IEC61400-25 har til et sådan system. Fra disse krav kan en generisk prototype bygges som en vindmølle kan anvende ved hjælp af et hovedinterface som benytter web services. Prototypen består af flere uafhængige moduler, som tillader fuld frihed i opsætningen for slutbrugeren.

Opsætningen af systemet skal være let tilgængeligt, således at slutbrugeren kan tilføje eller fjerne moduler efter afviklings krav.

En prototype er blevet udviklet med det formål at undersøge disse aspekter.

Preface

This thesis was prepared at Informatics Mathematical Modeling, the Technical University of Denmark in partial fulfillment of the requirements for acquiring the M.Sc. degree in engineering.

The thesis deals with the different aspects of creating a generic server for a common wind power plant standard. The main focus is on evaluating possible solutions that can be used, in order to achieve a good and flexible system, without compromising the final requirements for the whole system and the different modules it is comprised of. The final system has been designed with configuration in mind.

Andreas Kargård Olsen
Baris Ozdil

Papers included in the thesis

Proceedings paper for 2nd International Conference on Integration of Renewable and Distributed Energy Resources

Acknowledgements

First of all we would like to thank our main supervisor Bjarne Poulsen and Knud Ole Helge Pedersen for many insightful conversations during the development of the ideas in this thesis, and for helpful comments on the thesis. We would also like to thank the following persons for sharing their knowledge with us.

Knud Johansen Q-TECHNOLOGY, IEC Gamesa,
Claus Bjerge E2 DONG Energy,
Aksel Kargård Olsen,
Jacob Østergård.

INDEX

Chapter 1

INTRODUCTION

In this chapter an introduction to power plants and the arguments behind the need for a common communications standard will be presented. The IEC 61400-25 [IEC] standard that is used for communication with a power plant will be described. The visions for the project is presented together with a project description, this includes defining the overall scope for the project.

Wind power plants have through the years steadily gained a bigger and more dominant position in the power generation industry. Each vendor has their proprietary solutions on controlling and monitoring of the products supplied. In today's ever changing and rapidly growing energy market, monitoring and easy communication is essential. Through this communication the current state of the individual power plant can be controlled and monitored when required, and counter measurements can be enforced if needed in order to supply the demand for energy and keep the stability of the distribution system. It is vital that the overall dispatching systems are able to control the energy generation from a wind farm on demand in order to meet the fluctuations in the energy consumption. A common way to achieve this is a vendor independent approach.

1.1 Historical view & Motivation

Since the early days of electric power, distribution, standardization and integration has been an important issue to overcome the complexity and manageability of power networks with many sometimes multiple providers and many consumers. Historically, many networks were composed of huge energy suppliers like coal fired power plants and nuclear power plants which excessive production capability in order to secure that peek energy demands could be fulfilled. Today the energy production facilities are composed of various distributed generation units with different capabilities For example the output of the wind farm is dependent on the strength and frequency of prevailing winds, whereas the burning of fossil fuels is dependent on their availability and price. In today's competitive market, excessive production capability is not economical favorable due to the running cost on the spare capability and related environmental issues. This includes the overproduction of unnecessary pollutants. However, fixing the production to an estimated average consumption does not solve the problem. Transmission bottlenecks will appear, as the electric power consumption increases and the delivered electric energy does not meet the demand. Bottlenecks will increase the risk of blackouts and can in the worst case lead to a total collapse of the entire power grid. This happened in the North Eastern United States in the summer of 2003 where millions were left without power, and key infrastructures were shut down. Another vital part of preventing these blackouts according to José Delgado [JDAT03] is:

"For a standard set of actions deemed necessary to maintain the stability (reliability of supply) of the power grid and the entities responsible for those actions."

What this means is that a standard way to communicate essential parameters and related issues is needed.

In order to maintain a stable power grid with adequate capacity when needed, various power generating entities must be able to regulate their production. The regulation should be done in response to other entities in the power grid or to the power system as a whole. What is rational for one entity may not be so for the complete system or the energy market as a whole. Distributed power generation systems become highly complex and calls for careful regulation.

As the complexity of power distribution network increases, methods for efficient analysis, monitoring and coordination of the network control becomes very essential. This in turn demands a highly efficient and dynamic control strategy for the power system network. Several organizations have addressed this issue in manners where the main objective has been to develop communication standards for interconnecting electric power generation systems.

Enabling efficient and stable communication between power generation units is not the only major reason to develop communications standards. Besides the historical motive, another important aspect is to obtain a vendor independent communication. This of course would cut the price for maintenance and management, meaning lower total cost of energy generated. [IntelliGrid]

One important effort towards standardization has been launched by the IEC (International Electrotechnical Commission)[1]. IEC is the leading global organization who prepares and publishes international standards for all electrical, electronic and related technologies. Their focus on communication within substations has led to the IEC 61850 communication standard. IEC 61850 is a standard for information modeling, information exchange services, and a mapping to MMS and GOOSE protocol stacks applied for monitoring and control of substation components. Several standards for distributed power generation are under development using the structure of the IEC 61850 standard. e.g. splitting the information modeling from the information exchange services and the transmission given by the mapping to protocol stacks. [IEC],

The IEC 61400-25 standard is based on the same basic structure and are focusing on the communications needs for wind power plants, which is one of the base elements for this thesis project. Many of the major vendors are involved in the process of developing the IEC 61400-25 standard where Energy E2[2] and Gamesa are among them. Both Energy E2 and Gamesa are involved as a business partners for this thesis project.

1 UCA has done some similar work, but the projects has been combined under IEC.
2 Energy E2 is a part of DONG energy.

They are both active contributor of the standard which is still under development and close to be finalized.

1.1 Vision

Energy E2 wants research conducted for a software system to be designed for monitoring and control of wind power plants based on the IEC 61400-25 standard. Currently no system has been developed complying with the 61400-25 standard except for a rudimentary demo system. The current demo system is only implementing a subpart of the standard without processing any actual data but only exposing a minor part of the web service stubs.

The goal of this thesis is to evaluate the standard and end up with a working system implementation, covering the major parts of the standard. By doing this essential work on evaluating the standard, and using it in a real world system, it will be possible to spread its use and make it become a widely accepted industry standard.

1.2 Project Description

In this thesis a prototype of a system are to be developed relying on the IEC 61400-25 standard which describes the information and the information exchange models and the mapping to protocol stacks. Wrapped around this prototype are several other applications servicing the system and making it usable. Evaluation of different solutions must be made during the design of the system, describing the approach used, and why it was used.

The system developed is comprised of the following major parts which are illustrated in Figure 1

- A Server process communicating with a wind power plant gathering data and storing the data. The information model used is specified in the IEC 64100-25-2 describing the conceptual information model for the system. This server process will act as a bridge between the data source (power plant) and the database.

- A Server acting as an interface to the database with proper functions in order to alter and communicate data. IEC 64100-25-3 defines the information exchange model used to communicate the data through corresponding services. A Major task for this server is to exchange data through the specified model, and keeping track of who needs what, and when.

- A server exposing web services described in IEC 64100-25-4. This is the actual realization of the data communication where the information exchange model is mapped to a communication profile.

- Client application to consume the web services and demonstrate the functionalities exposed by the server.

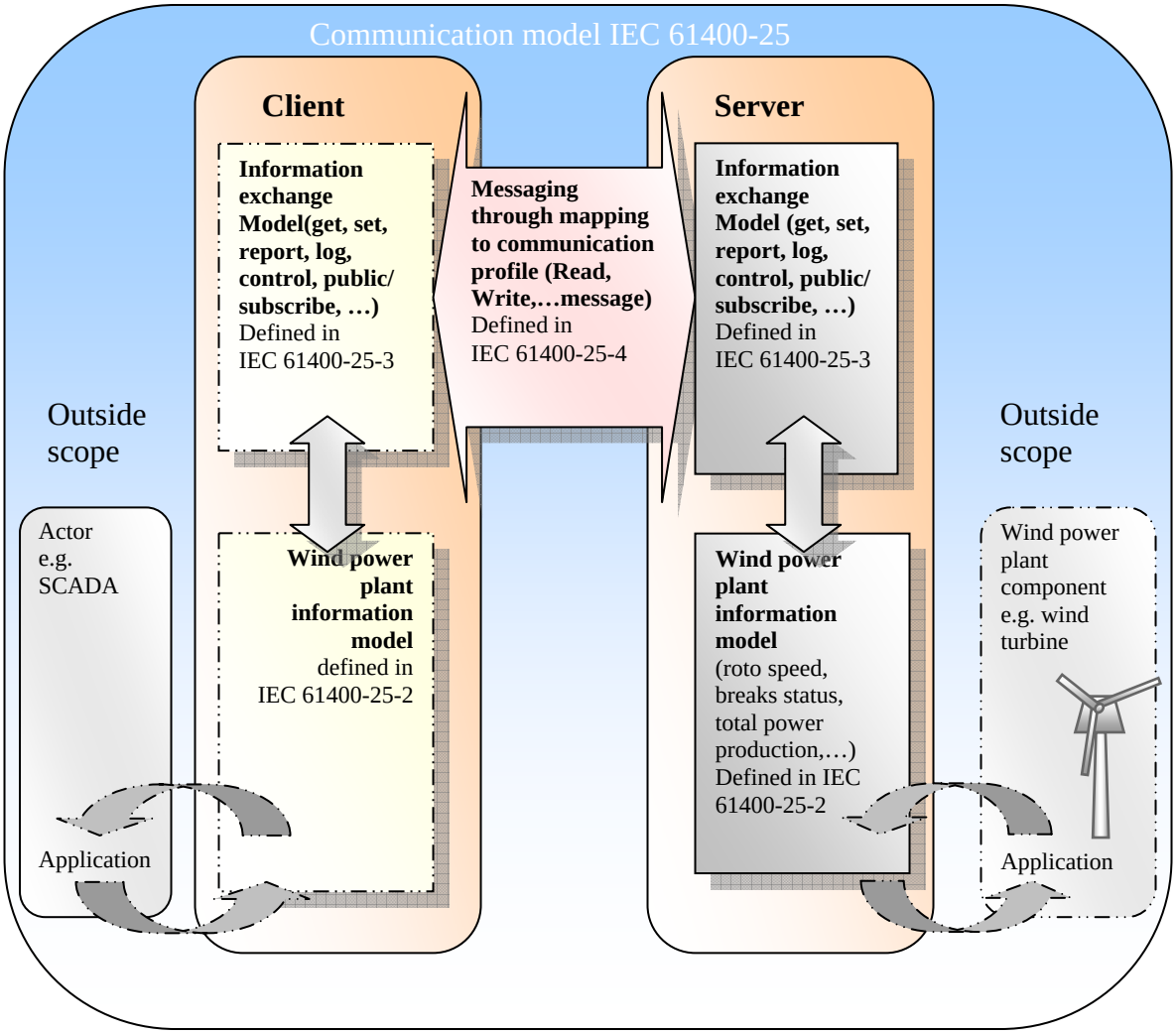


Figure 1
Communication model according to IEC 61400-25

The system must make use of web services as a communication technology together with a suitable architecture which in this case is Service Oriented Architecture (SOA).

In developing this system there are a number of important issues which must be addressed. The concepts introduced by the IEC 61400-25 standard must be investigated carefully in conjunction with real implementation techniques and technologies. This will help designing the system and make it possible to end up with a suitable reusable architecture The aim is to develop the system as a generic, configurable and extendable framework, abstracting away the complexities from the users and presenting the functionalities in an easy to use manner. To achieve this, the entire system domain must be analyzed very carefully where afterwards a fine grained solution to all issues must be provided. In the next chapter a detailed analysis of the important parts of the system will be provided.

The following list summarizes the major issues which must be addressed and solved within the scope of this thesis.

- **Evaluation of the standard**. Whenever the standard proposes ideas or suggestions there might have some implications, unwanted side effects, or would have a better alternate solution, an improved suggestion has to be presented, describing the details the implications of implementing the change(s). Whenever possible, the solution should conform to well-known software design patterns.

- **Making a framework encapsulating the system as a configurable module**. Even though a standard communication protocol would make it easier to communicate between wind power plants from different vendors, the success of the standard is still measured by whether it is easy and feasible to use. A framework encapsulating the standard must be designed. This framework should expose the methods offered and taking care of all communication. The meaning of trivial means that the framework invokes the different methods. On top of this communication is added security, reliable messages or what ever is needed for the system. The framework should be transparent, meaning that a user can change the configuration of the framework. The approach for adding different layers to the communication must be done on top of the communication model, to ensure that different approaches can be used, for instance to secure communication, and the framework will do most of the work involved in the transition automatically.

- **Service Oriented Architecture (SOA)** This kind of system is comprised of many different devices, distributed over a great physical space and requiring a variety of functionalities. SOA is for one a system architecture where a network is comprised of nodes exposing services to each other in order to complete at greater common task. The system will take advantage of this architecture whenever suitable.

- **Security**. The aspect of security is a major part of a system communicating data over the network. This includes,
    - Reliability (a system you can trust, in all aspects.)
    - Confidentiality (information is only shared between the right people.)
    - Integrity (The information is guaranteed to stem from a specific person and has not been altered by a third person)
    - Availability (The information is accessible whenever it is needed. This also include aspects where the system is able to recover incase of a failure, and bring it self back in an operating state as smoothly as possible.)
    - Non-repudiation (Method by which the sender of data is provided with proof of delivery and the recipient is assured of the sender's identity, so that neither can later deny having processed the data.)

    The system must address the security aspect making sure that the listed rules are satisfied.

- **Configuration.** For the time being there has not been specified a common way to configure the system or its constituting devices. In the future an extension to basic parts of standard will be released describing a configuration language. The system must be designed in a way where everything can later be configured through a common interface. Included in this topic are also the problems of managing and maintaining the system.

- **Data handling.** A massive amount of data is collected. This data has to be analyzed, filtered, and made accessible to clients as fast as possible. This leaves a lot of constraints on the storage device according to access time, storage size and stability and so forth. In reflection of this, a great deal of work must be put into the design of the storage structure making sure that it can meet the specifications that a system like this demands. The Main focus being on fast data retrieval for the clients.

- **Clients.** The system should include a proposal for client systems.
    - A web client being able to display data
    - An application client. Not all of the functionality the system offers is suitable to display in a web based client. This includes the possibility for the client to subscribe to a service. (The system must keep track of all subscribing clients sending data as it occurs.)

1.3 Time Table

Please refer to Appendix L.

1.4 Prerequisites

To run and understand this thesis there are a number of prerequisites that has to be fulfilled.

Installed software:

Net framework 3.0 runtime components and/or Microsoft windows SDK
Internet Information server, or other server capable of hosting WCF web services.

Basic knowledge of:

- The C# programming language.
- XML Web Services and Windows Communication Foundation. (WCF).
- General knowledge on system design and software development.

1.5 Report Outline

This report is organized as follows.

Chapter 1 – Introduction

Provides an introduction to the project and explains the background. It includes a project description together with an overview of the system.

Chapter 2 – Analysis and design

Provides an analytical discussion based on the standards and architectural properties of the system. A major part of this chapter will be of describing different possibilities for solving the tasks. It is mainly divided into three sections described below.

1. The analysis of the standard and models described in it.
2. Provides system design possibilities based on the analytical discussions. The design will form the basis for the implementation.
3. Propose a conceptual model for the system based on the previous analysis. This will be the basis for the implementation for the prototype

Chapter 3 - Service oriented architecture and standards used

An analysis on Service Oriented Architecture (SOA) in conjunction with the standard will be presented. A detailed discussion of Windows Communication Foundation (WCF) is presented.

Chapter 4 – Case study Implementation and Test

Gives an overview of how the system is implemented with what technologies. The solutions to key problems are discussed including security aspects different client system and on/off communication.

Chapter 5 – Conclusion and future work:

This chapter provides a chapter wise summary of the project, A conclusion of what has been accomplished, and the ideas for future extensions and improvements to the system.

Chapter 2

ANALYSIS OF THE STANDARD

This chapter can be categorized into three parts where in the first one, a description of the IEC 61400-25 standard, will be given in order to give a clear view of its scope and the general problem domain, and what the requirements in the domain are.

In the second part, the main problems and challenges are identified and described in detail. The approach is to give a clear analysis which outlines the important problems of the system being implemented.

The last part will propose a solution addressing the identified problems. The proposal will suggest a system architecture containing the solutions for the identified problems, in a platform, and language independent manner.

2.1 Analysis

The IEC 61400-25 series is a specialized version for defining and standardizing a unified communication for monitoring and control of wind power plants. The aim is to enable systems from different vendors to communicate mutually.

The IEC 61400-25 series is an extension of the previous IEC 61850 series of standards which in general defines the communication networks and systems in substations. IEC 61400-25 does not simply replicate IEC 61850 but reuses the definitions which in general apply to all power systems. IEC 61400-25 extends the IEC 61850 by defining specialized information models in order to describe the specific Wind Power Plant components such as rotor, turbine and the like.

The view and approach of the standardization expands over the information modeling of the target system and the communication protocol for communicating the data encapsulated in the information model. As a result of this approach, the standard addresses the domain by separating it into three main categories of interest which together encompass all the important aspects of the communication and control of wind power plants.

The three different categories are as follows:

1. Wind power plants information models.
2. Information exchange models.
3. Mapping to communication profile.

In these series the information model is the base for defining the communication related aspects of the standard, since it should be possible to communicate the data instances of the types defined.

The IEC 61400-25 mainly defines a client-server architecture with the three aspects mentioned above. This is illustrated in the figure below.

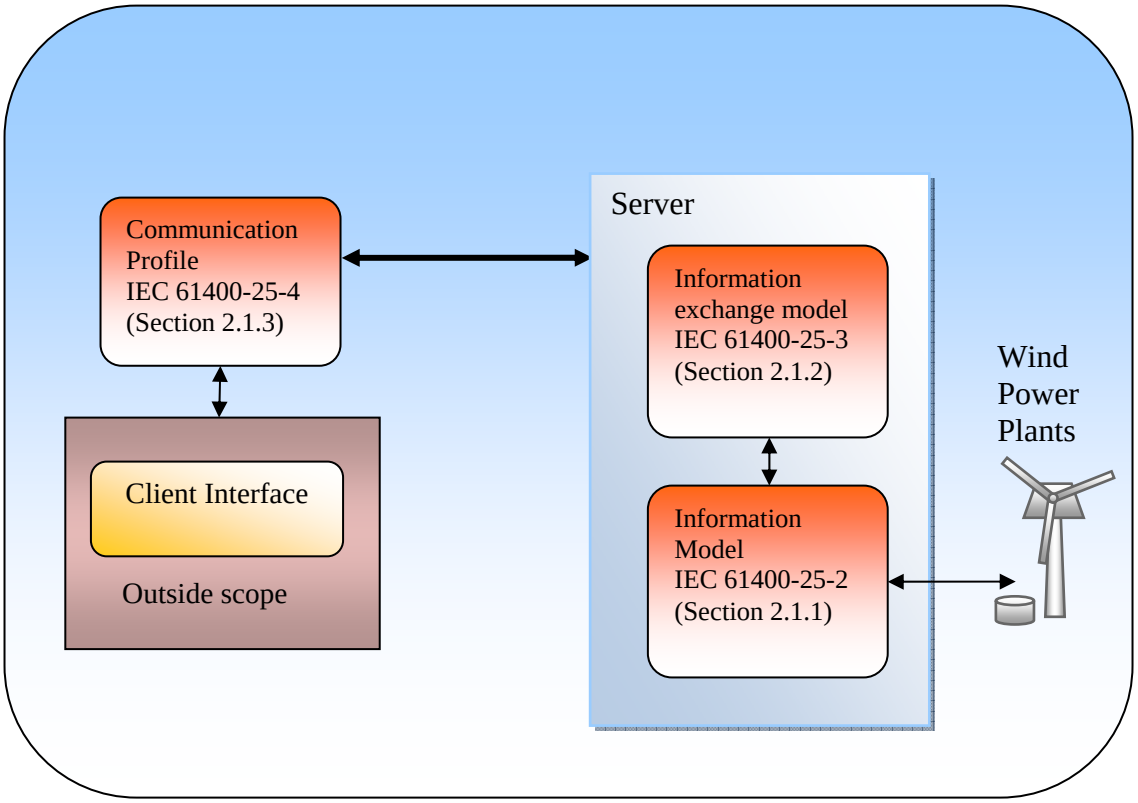


Figure 2
Communication model defined in IEC 61400-25

2.1.1 Information Models – IEC 61400-25-2

The information model makes use of an Object Oriented approach for modeling the wind power plant components and data in general. It defines an exact model of the wind power plant (WPP) which contains the components and data of interest. This data will be made available to access for monitoring and control purposes. The information model constitutes a precisely defined set of reusable data classes which will make it possible to build up a logical model being able to represent a specific WPP which is to be modeled.

The WPP is abstracted as a logical device encapsulating different types of classes being able to represent different parts of a real wind power plant device. The data classes defined to represent parts of a WPP is grouped and located under units named logical nodes where each node represents a component on the real WPP device. This will make it

possible to group related operations on data closely related to each other. For instance, the wind turbine residing in a WPP is modeled as a single specific logical node where it will be possible to embed all data and related operations in it. Closely related data will most probably reside on the same logical node. This will make it easier to carry out operations on related data since the data is located in the same container so that data referencing is not complex and thereby making it easy to fetch it. It will be easier to implement and to capture an overview of the device being modeled because its logical structure closely resembles the real world. The sections below will go through each modeling concept used in the information model.

2.1.1.1 Logical Device

As stated previously, a logical device is defined as an abstract model being able to fully represent a power plant device, which is in IEC 61400-25, a wind power plant. The Logical Device (LD) encapsulates all the logical nodes necessary to contain all the data for the WPP device.

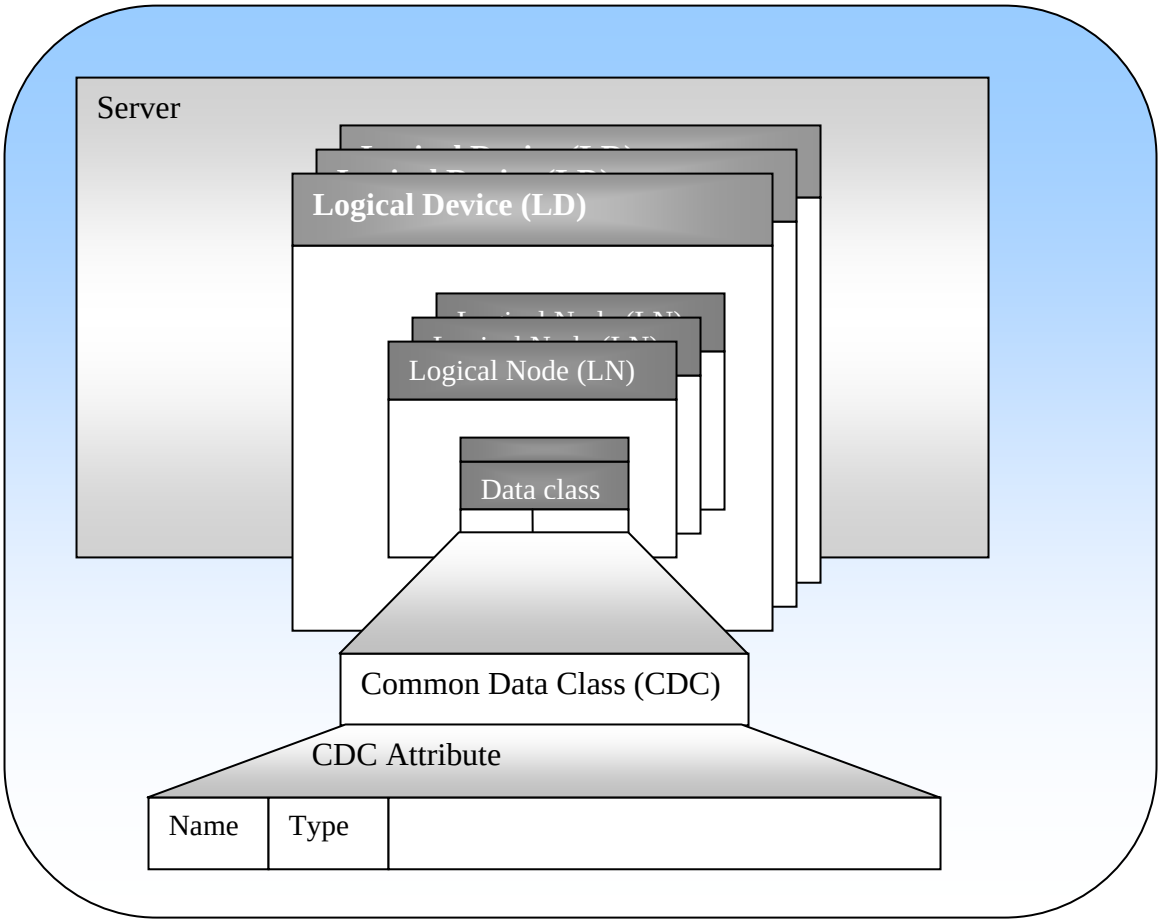


Figure 3
Information model hierarchy

1. A LD residing on a server is assigned to a specific WPP which must be able to fully represent it with all its data and attributes. It must contain a collection of specific Logical Nodes (LN) which will further contain data instances reflecting the physical state of the real WPP.

2. Beside the Logical Nodes specific for a WPP, a logical device also contains data about its own condition and the physical device (server) that is hosting the logical device. This common information is contained in specific Logical Nodes serving only for this purpose. For instance, the logical node LPHD (Logical node physical device) represents common data of the physical device hosting the logical device. These logical nodes are described in Clause 8 of IEC 61850-7-1

Briefly, a Logical Device is a container for all WPP related data together with self descriptive meta-data describing the physical host and the device itself. The hierarchical structure of a logical device residing on a server together with logical nodes and data classes is depicted in the Figure 3 above.

The Figure 4 below illustrates the relation of the model to the real world.

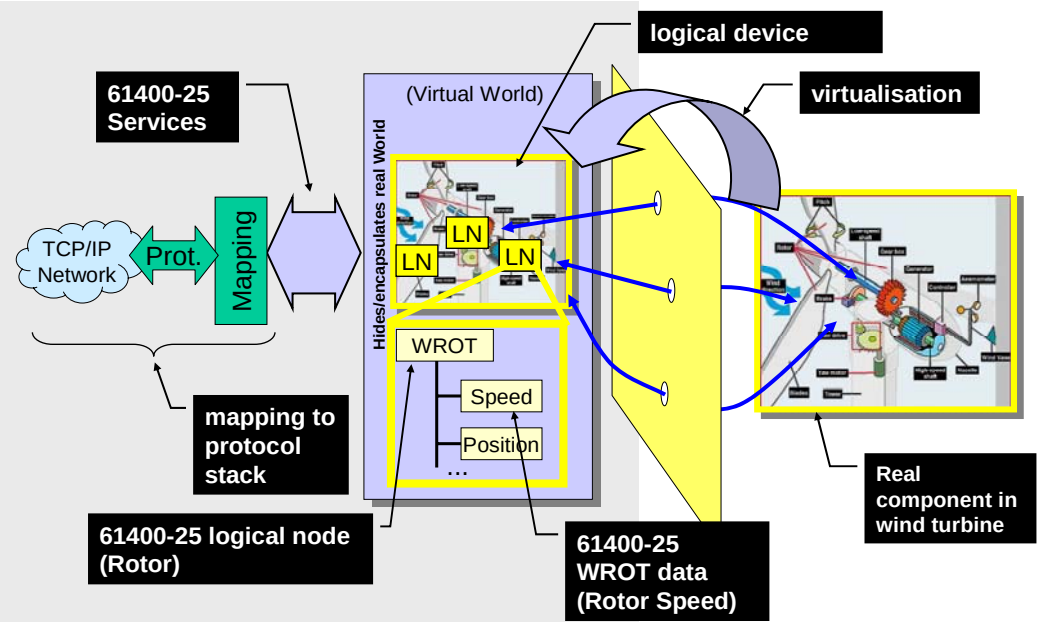


Figure 4
Modeling approach (IEC 61400-25-1)

2.1.1.2 Logical Node

It has been mentioned that a WPP is decomposed into smaller information units where they are distributed to different containers. Each such container is defined as a logical node (LN). The modeling approach uses the logical nodes as building blocks to construct all the information of a WPP which is subject to a service. The modeling approach is stepwise decomposition and composition for WPP and the information model respectively. An example of a logical node could be WTUR[3] which represents the general information of and wind turbine. This logical node encapsulates information such as net active energy production and net reactive energy production.

1. A logical node consists of a collection of related data, defined as data classes (DC). All the information in a logical node is contained in respective data classes. The structure of all logical nodes is similar and has a standardized form where different types of logical nodes can be constructed through the combination of different optional data classes they may contain.

All the logical nodes used in modeling the WPP inherit their structure from the abstract logical node class defined in IEC 61850-7-2. From an implementation point of view the implementation of these different logical nodes will be similar since the structure is based on a common definition and follows a common pattern.

2.1.1.3 Data Classes

The data class is the actual component of the information model which is used to define any data contained in logical nodes. It is a generic template to define specific data classes which encapsulates the information. These specific classes are called common data classes, which are precisely defined classes inherited from the general data class. These classes are defined according to similar needs for different systems in order to have a reusable data class repository. The common data classes used to model a wind power plant device can mainly be categorized under two groups. Common data classes defined specifically for wind power plants and logical nodes inherited from IEC 61850-7-3. A complete listing of these common data classes is provided in IEC 61400-25-2 clause 7.

1. The level of data classes is where the information exchange services operate when retrieving real WPP data. The values of data instances can be written and read depending on their configuration.

2. Similar to logical devices and logical nodes a data class instance must have a unique name among data instances at the same level in the information hierarchy. This will make it possible to have a unique path from the top level logical device to the instance itself. Therefore each data class has a data reference attribute making it possible to reference it from the top level instance of the information

3 Wind turbine general information. This is information about turbine status, power generation and total net reactive energy production. What the node contain depend upon the specific vendor.

hierarchy. This is important since there is no other way to specify which values on the server/WPP one will retrieve.

3. Each data class has a set of data attributes which are all related to store and describe the data in various means. This includes the type, functional constraints, explanation of the data and trigger conditions that may be associated with the data. Each of these attributes will be used to model the data specifically as needed to reflect the actual data on the wind power plant device

4. The server must be capable of representing the information model with all its instances from logical devices and all the way down to specific data attributes. The hierarchical structure must be preserved so that each data instance can be referenced in a standard manner as defined in the information model.

5. A server may host one or more logical devices depending on the number of WPP which must be controlled by that server. Therefore it must be possible to uniquely reference a specific logical device representing a specific WPP. Each logical device must have a unique reference/name within the server/namespace it resides.

6. The server must be capable of representing a logical node contained in a specific logical device and to refer to it uniquely. Each logical node has a unique reference name which makes it possible to locate it directly in conjunction with unique reference name of the logical device containing the logical node. This is important since a single server instance can have more than one logical device residing on it. This means that the server can have different logical devices containing logical nodes with identical names. The naming convention used in the standard prevents unambiguous references to any device or data instance.

The data model is defined in an object oriented manner, and must be reflected as such in the data Model.

2.1.2 Information Exchange Models – IEC 61400-25-3

This part of the standard describes the information exchange model which is implemented on the server enabling client's systems to access and modify data in the information model. Each information model instance has a service interface describing the operations available on that particular instance. The services described in the information exchange model are designed according to the specific information model instances. Each information model object has a specific set of services making it possible to read or write from/to it. For instance, a logical device instance is associated with a service *GetLogicalDeviceDirectory* which will retrieve all the references to logical nodes contained in the device.

2.1.3 Abstract Communication Service Interface (ACSI)

The basic services that are used to mediate between the outside world and the real wind power plant device are referred to as Abstract Communication Service Interface (ACSI). The basics of these services described in details in IEC 61850-7-1 and IEC 61850-7-2. The figure below illustrates the various components of the ACSI models.

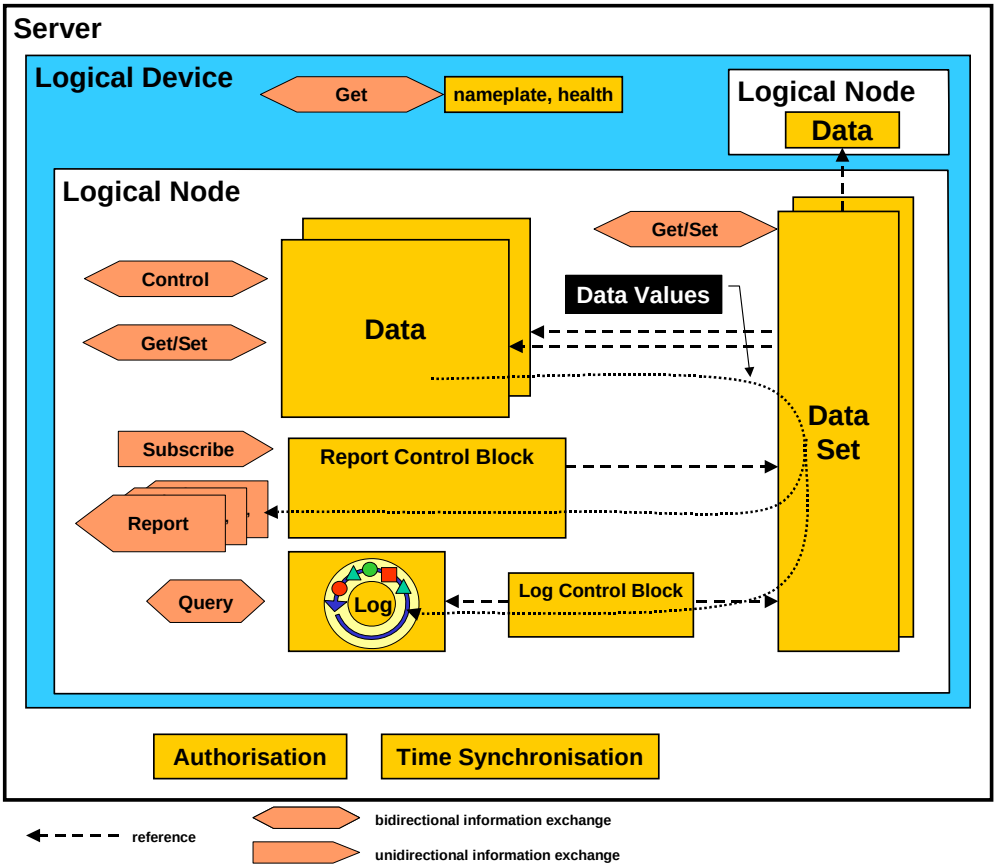


Figure 5
Information exchange model for wind power plants (IEC 61400-25-1)

A physical device with a communication interface is represented on a server as a logical device. The server is accessible over the network making it possible to accept client system connections and support services to the client following authentication.

The logical device(s) contain(s) logical nodes that represent the various components and functionality of the logical device. Logical nodes can respond to control inputs, provide reports configured by the client, and contain logs which can be queried by the client. The logical nodes contain data objects which can be written to or read. Read and write operations are done through the services provided at the level of the logical node and downwards. The client can create data sets of an arbitrary number of data objects and read or write to these data objects as a group as well as being as being able to read and write individually named data values.

Services are provided which enables a client to create, delete, list, read or write to data sets. This arbitrary grouping lets the client system define collections of data attributes that are commonly needed and retrieve them with a single read operation using a single name referring to the data set.

In the ACSI models, the information that gets reported or logged is represented by data sets. This permit specifying the rules for reporting and logging to be defined in a more compact manner applying to a group of data.

2.1.3.1 Reporting

The reporting services must make it possible to subscribe to spontaneous data reports on specific conditions for data values. Conditions such as change of value or change of attribute values will trigger a preconfigured reporting subscription and start dispatching the values. It should also be possible to cancel a report subscription.

1. In the reporting mechanism, the server must make available the data for the client to read and write. As mentioned above the data subject to retrieval can be configured by the client as a group of data objects named data sets. Since it is not common practice in client server architecture for a server to contact a client offering data, the server should buffer the values to deliver it later to the client, whenever a client request is made. Otherwise the server has to contact the client and deliver the data to it, which turn upside down the client server architecture.
2. In order to achieve buffering mechanism so that the server does not have to notify the client for the available new data, some sort of a server side session must be implemented for the reporting interaction between client and server. The server must have an internal state making it possible to keep track of which step in the reporting process it is in and which data have been sent and which is still in the buffer. However, the standard also specifies that it must be possible to configure the reporting mechanism such that data is not buffered in case of a connection interruption, meaning that the client only can access the data available at the time it makes its service request.

The reporting mechanism is controlled by an REPORT-CONTROL class which can be configured through its attributes. The REPORT-CONTROL references an instance of a data set which groups the data in interest. The attributes of an REPORT-CONTROL instance used to configure the reporting can be accessed through specific services made available for the REPORT-CONTROL class. The details of the REPORT-CONTROL class are defined in 14.2 of IEC 61850-7-2.

Buffered Reporting

A client process can configure the behavior of buffered reporting through an instance of a BUFFERED-REPORT-CONTROL-BLOCK (BRCB). This instance is created through the *AddSubscription* service made available to client processes. A BRCB instance has a set of attributes used to control the behavior of the reporting mechanism. It also has a DATA-SET reference for which the reporting is applied to. Whenever a BRCB instance is enabled an Event-Monitor should monitor all the data instances referenced by the report-control-block DATA-SET. The Event-Monitor must send event notification instances to the Report-Handler which will further delegate it to the respecting BRCB instance.

Whenever the BRCB instance receives an event notification it will generate a report and push it onto the report buffer stack, making it available for client retrieval. In case of a client thread blocked on a buffer event, the BRCB instance must notify the client thread to pick up the generated report. The BRCB should generate a report even if at that moment there is no client actively waiting for a report. The immediate delivery of reports to clients requires a fine grained synchronization mechanism which will be discussed later in this section.

Un-Buffered Reporting

A client process can configure the behavior of un-buffered reporting through an instance of an UNBUFFERED-REPORT-CONTROL-BLOCK (URCB). Un-Buffered reporting is similar to Buffered reporting with one important exception. As the name implies, in Un-Buffered reporting the generated reports is not buffered on a report buffer. If a previously generated report is not picked up by a client, this previous report will be swapped with the new report. That is, the order of sequence or the loss of reports is not considered by the server. A report can be retrieved by the client in the period until another report is generated, otherwise it will be lost.

Report Handler

A report handler object should be able to maintain all the subscribed REPORT-CONTROL-BLOCK instances together with the report buffers assigned to each client process. The dispatching of report requests should also be controlled by the report handler. It must be possible to enable a mechanism so that client processes can wait for a specific amount of time to retrieve a generated report. Immediate delivery of generated reports will be possible through such a mechanism which is required for an event driven data retrieval model.

In order to have a wait/block mechanism for requesting clients, the report handler must maintain a thread pool with reusable threads in order to make the process fast and efficient. Creating a new thread for each report request will use unnecessary amount of resources on the server side.

Whenever a client requests for a report, the report handler must dispatch an available thread to the client and make the assigned thread block on an event. By this, the report request service will not return immediately but instead wait until it is released and thereby return the response object to the client. Whenever a report is issued the report handler will notify the awaiting thread and release the blocked request so that it will continue its execution and thereby return available reports to the client if any. The wait/block mechanism must have a suitable wait-time and release the block so that the underlying service connection does not timeout. The two different cases in wait/block mechanism are illustrated in the figure below.

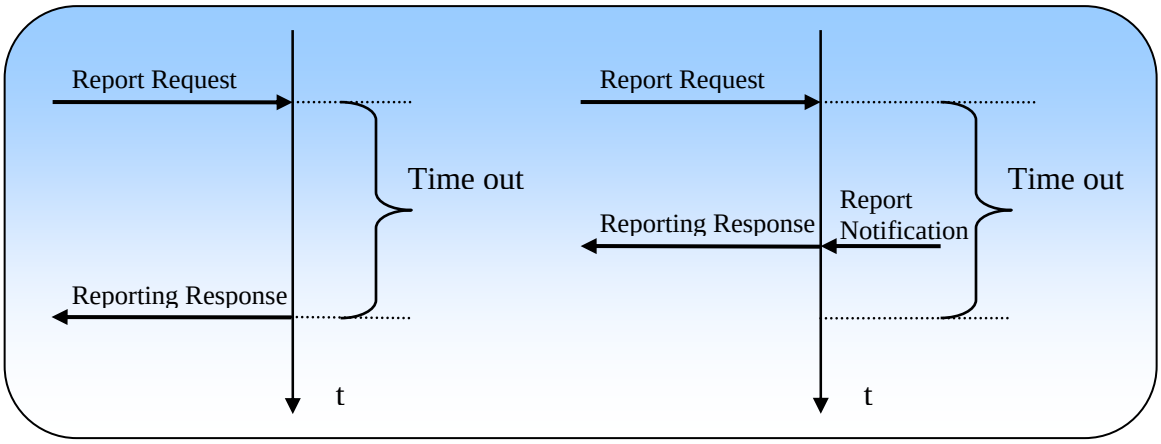


Figure 6
Wait/block mechanism

Each client is responsible of reconnecting to request and wait for new reports after a report request service call has returned. The wait/block mechanism is a polling mechanism extended with timeouts to achieve a similar behavior as in a connection oriented interrogation. It is not possible to establish a pure connection oriented interrogation when using communication technologies such as web services.

Event Monitor

Monitoring events is an important part of the reporting mechanism. It should be possible to monitor the data instances referenced by RCB instances (through DATA-SET) configured for each client. That is, only the data referenced by reporting will be monitored. Whenever the state of a data instance alters, the event monitor must send an event notification to the report handler. The event notification must contain a reference to the subject data together with an indicator of the type of state change.

After notified by the event monitor, the report handler must identify the RCB(s) referencing to that particular data and generate a report for each of the RCB. Depending on the RCB the generated reports will be pushed to the report buffer or delivered directly to an awaiting client.

2.1.3.2 Logging

Logging like reports can be initiated upon the client's request. Like reports the data can be logged on the same criteria. (Change of value etc.) But in addition to this, updates also must be logged.

The client must at any time be able to get a log stream for a given interval. Due to the data storage amount it must be assumed that this time storage is limited concerning the log available for the WPP itself. Reports reflect current data while logging reflect longer term data, and system status.

2.1.4 Mapping to communication profile– IEC 61400-25-4

The services defined in the information exchange model are mapped to standard web services. A detailed description of each service is provided together with the corresponding WSDL document describing the exact structure of the service methods. Each service defined for the various data models is mapped to SOAP services making it possible transfer data with the correct types and structure defined in the information exchange model.[WSA2004][IEC61400-25]

1. The server must resemble the web service description provided in the communication profile mapping. It should be possible to communicate between client and server accordingly as specified in the information exchange model.

2.2 Challenges and design requirements for the system.

In order to develop a good design for the system, it is important to focus at what is required to build a stable and efficient system according to IEC 61400-25. A presentation of the challenges in designing and implementing the system will be presented next, together with a suitable solution strategy addressing the identified problems.

The following topics have been identified for the design:

1. **Module based components.** The standards does not provide a solution for every single aspect in implementing the system, and in many places the standard leaves several choices or decisions for the system designer to solve. The communication with the client can be done in several ways, using web services or MMS etc. The standard proposes a solution for some of the challenges, but it is only a proposal, and therefore there are no guarantees that a vendor will use it. What the vendor uses is totally independent. For instance security or optimization of the system might not be solved in the same matter for every vendor. At the same time the standard is "a work in progress", and will most likely change in several places. As a consequence, the system must be divided into easy replaceable modules. Combining these modules will make a framework, which can be reused to make development and further extensions easier.

2. **Configuration.** The information model defined in IEC61400-25-2 has a lot of optional nodes that can be implemented with optional data classes. The vendor must have a way to configure what their WPP has implemented, and how the communication is done. It should be easy to configure the server when deploying when constructing a logical device to be made available for access. Providing such a configuration option will make reuse of components much better and efficient and it will be easy to set up a server being able to represent specific wind power plants.

3. **Data storing and handling** Lots of data is flowing through the system. To make a system which can handle the massive amount of data flow it is vital that data can be processed in an efficient and secure way. The data must also be rapidly retrievable and easy to store.

4. **Security.** Security is an important aspect of the system. Depending upon who you ask you will get different answers to the question of what security is. The most common security topics is concerned with
    a. **Data**. Data is transported safe, and its integrity is protected.
    b. **Personal access and management security.** Who can access what and perform which operations.
    c. **Reliability**. Can the system provide the service that it is supposed to when it is supposed to?

For an easy to configure, secure and reliable system, each of the topics above must be addressed. The following section will go through the more specific details.

2.2.1 Module based components in a framework.

No matter how ingenious a design is, it still rises and falls depending on whether it is easy to deploy. By designing the system as a framework all the major and trivial tasks of the communication according to the standards must be encapsulated. The framework must take care of all the interaction with the protocol. And all the quality of service aspects, security, transactions, sessions etc. must be implemented in a standardized way. The framework must have a default setting for communication, but it should still be possible to change the different settings if needed. Looking at end to end communication, there are many single elements that must be easily changeable. For instance each WPP might have its own way to supply the data. The framework must be flexible. To achieve this all the major parts of the framework must be placed in a separate dynamic linked library (DLL). The system must be a very-late binding architecture, where one component's runtime is integrated with another component's runtime using dynamic invocation. The components can then at runtime determine which components of a specific class are available to it, and be independently able to load and use them, through reflection.

The system is to be designed using multiple modules that are to be changed on demand. This implies that a common interface must be used. No matter how the mapping in the transport is done, the resulting response must be the same. By designing some standard test cases that looks at what goes in to the system and what comes out, the system can test itself by running these tests. A Unit test does just this. Unit tests are written before or during developing of the system, where the functionality of the systems is in focus. The test frame should always be kept with the system, making it possible to run the test at any given moment. By having a good test frame for the system, errors introduced as a result of updates or legacy coding will be minimized. Changing device connector to a WPP can be reduced to merely replacing a DLL in the application folder.

Proposal

The system must be module based, where each module is located in its own DLL. A default setting must be supplied with the system, and testing must be done in a uniform way.

Pros

- Easy to change access interface for the system.
- Easier to maintain.
- Possibility for reuse.
- Makes uniform testing possible.

Cons

- Adds overhead to the system.
- More work at design time.

2.2.1.1 Configuration

The job of the vendor is to customize the system configuration so it works with the specific WPP. This should be done as simply as possible. The configuration contains information about what in the information model (IEC 61400-25-2) has been implemented by the vendor.

Configuration also includes information for how the interaction between the WPP and the system must be performed, for instance how often can/must the data be pooled from the WPP; or are there going to be used any security aspects for the communication. There exists three major discrete moments for configuring the system.

- Design time
- Deployment time
- On the fly.

Design time

The configuration information can be put into the system at design time. The data is entered into the program code for the system. If the system is not going to change over

time, this would be the ideal way to do it (static data). Entering the information directly in to the code would result in the system needing to be recompiled and reemployed if any changes are to be introduced.

Deploy time.

The information about implemented nodes does not change for a WPP over time. When the WPP has been installed there is no need to change the static information about the system. Even if the system administrator wants to change what is stored, it would still be the same data being pooled from the WPP. This indicates that the data could be entered at design time; however it would be a better solution if the system did not have to be coded specifically for each WPP. If the WPP provide the system a configuration file, the system could set up itself when the WPP is installed. The application logic would be the same for two different WPPs but the configuration file would be different.

On the fly

A third way to configure the system would be through dynamic storage. The time between data pooling must be configurable in a fast and easy way. The same applies for administrative information, like access rights. By adding the ability to change configurations at runtime, the system is ensured to be flexible. It is not plausible to have every configuration variable as a change on the fly variable. First of all, the system has to have some logic to handle situations when a variable changes. If the variable is not very likely to change there is no reason to spend a lot of time to make such a change possible.

Proposal

The information model does not contain information of how the WPP and the system will communicate. It only states an abstract data format. Each vendor would have to design an entire system from information model to exchange model. How data is stored and how it is gathered it closely related. It is impossible to design autonomous interchangeable modules as long as a common way to transfer data from the WPP to the system does not exist.

The configuration must be done in two different ways. The configuration of the WPPs implemented nodes is placed in a file. The first time a WPP is turned on it will configure the system for that WPP; setting up data storage etc. This ensures that a module can be reused without a lot of extra programming. Configuration of user rights and manageable options (Access Control) must be configurable similarly through XML configuration files.

Pros

- Easier to change and reuse the components of the system.

Cons

- A standard has to be defined of how the information is to be handled, so it must be designed from the bottom.

- No standard yet, and it will probably change in due time.

2.2.2 Data processing and data storage

On the server side, the information model must be represented reflecting the exact hierarchical information structure. It is extremely difficult to represent the model using a classic relational database. Besides the difficulty in representing the model, data retrieval will also be extremely slow. This is because the server must execute a large number of queries to retrieve data from the complex relations in the data hierarchy. Adding this overhead to each client request will result in the server making database queries most of the time when serving client requests, resulting in slow response times and low throughput.

The complexity stems from the very nature of the WPPs components, not from the choice of design technique being used in the information model. The information model represents an object oriented model of the wind power plant which is a different paradigm from that of a relational database. There is a mismatch between the models used to represent information in an object oriented model and a relational database. This problem could be overcome by using an object oriented database instead of a relational database. Even so, performance and system throughput will remain a challenge.

Instead of storing the current data on a persistent storage for service retrievals, the information model could be constructed on the server process itself making use of the representational power of an object oriented language. The server process could programmatically replicate the information model and store the data in its run-time with suitable objects while running and servicing the client systems. This approach will make the execution very fast since the data is already stored in main memory ready to be fetched. And the modeling/representation of the information will be straightforward through instantiating objects from class definitions resembling the class definitions in the information model.

Each logical device made available for access on a server must be constructed hierarchically reflecting the structure given in the information model. Since each logical device could have different configurations because of different wind power plants they each represent, it must be possible to configure the way the server process instantiates these logical devices. When building logical devices, this approach will make the server able to use the full power of the representational capabilities provided by the information model.

To store the most current data and make it available to client processes, the server must continuously interrogate with the WPP for new data made available. The data must be swapped from the logical devices in the server process to a simple database to make room for new data. The database used in this case need not be a complex database reflecting the exact structure of the information model. The values could be stored together with their object reference where it later will be possible to deduce where the

data belongs in the information model. This database can be as simple as possible, since it will not be used to store data that will be needed by the server process to service the client requests all the time. It will only be used when retrieving data within a designated time window. Most of the data needed by the server to perform the services it provides will be stored in the server process itself.

Proposal

Only the current latest data is kept inside the system to speed up data handling. Long term data is separated from the running system to cut down runtime challenges. The information model is modeled in an object oriented way, to keep the implementation and the information model close together.

Pros

- The object oriented model applies directly to the information model.
- The internal data handling can be done in a fast way. Only current data has to be searched through.
- Splitting short term (current) and long term data handling.

Cons

- Construction of the internal data model (on the server) to represent the information model can be programmatically complex.

2.2.3 Security

There is no doubt that security is an important issue to be addressed. IEC 61400-25-3 defines the security aspects for the standard and how to solve it in general, but how it is handled specifically is completely up to the individual supplier. One supplier might simply use a secure line, and therefore remove any security aspects of the service itself, while another might want to use a public ISP where the service related traffic must be secured by the service itself. This calls for a solution where security is built on top of the communication as a separate layer in a modular fashion easy to add, remove or change on demand. Since security is a requirement, a default security profile must be provided.

The security aspects for the system which must be addressed are as follows:

- The security modules must be easy to add and remove.
- Have a default security taking care of the most common security aspects.
- Security should not be based upon a specific system or language.
- Security must not be protocol dependent.

Even though no formal security requirement has been defined, IEC61400-25-3 does suggest some typical security aspects which should be addressed.

- **Authentication**. The client and the server can be sure that a message has been produced by a by each other, and not a third party.
- **Integrity.** The messages are guaranteed not to have been modified or destroyed by third parties (transactions).

- **Confidentiality.** Only authorized people can read the messages in the system
- **Non repudiation**. The client and server can not deny having participated in communication.
- **Reliability.** The system must guarantee functionality This includes assurance of delivery of messages and guaranteed reply
- **Authorization and access control**. Clients can on view allowed logical nodes and data through allowed service methods.
- **Availability.** The system must be available whenever the client needs the service.
- **Quality of service.**

The statements are related to different aspect of the system. The first one is security of data. The first 3 are the most important ones, and are commonly known as CIA

2.2.4 Traffic security

In TC 57 [FC] a proposal has been presented in which a security model has been suggested. In this thesis TCP/IP traffic is utilized, and are therefore to follow the IEC 62351-3 security standard. The standard recommend transport layer security (TLS) to tackle the most common security threads. At the same time it specifies that security must follow the progress and update to better solutions when available. Another possible security solution is to use message level security. Here, security is applied at the message level rather on the transport level. The next chapter describes the two different security levels.

In any case the security must be transparent for the methods sending and receiving messages. Both of these use public-key cryptography, and provide the same security features such as confidentiality integrity and authentication. In both cases you have the freedom to pick the features you want, and simply leave out those that are unnecessary for the system. Server and client processes, of course have to use the same security scheme.

Proposal

Proposing a secure system according to the standard, it is important that it can be applied and removed at whenever required. The security is optional and must be replaceable if another schema is better suited. In later chapters there will be a description of how security can be implemented via the use of security profiles.

Pros

- Easy to specify the security elements needed.
- Standard way of defining security
- Future security profiles can be used.
- Independent upon the underlying functions.
- Protocol independent

Cons

- Slower

2.2.5 Access control and management functionalities

Another aspect of the security is the access control. Access control has the duty of ensuring that only authorized people can gain access to the data. How and what the security includes is defined by the service mappings (SCSM)[4].
The information exchange model proposes a conceptual authorization model. This includes:

- Access control.
- Restrict access to class instances.
- Class instance attributes.
- Method restriction.

In IEC 61400-25 the minimum requirement for access control is only defined as the need for supplying a valid username and password to gain access. This ensures that only people with a valid password can gain access to the system. Different users might have the rights to perform different actions. For instance one user might only view data, and another might also write data.

The system only contains methods and data the client can access. The user either has the right to run a method or not. Through the method the client can either read or write data. Every client does not necessarily have the right to view all data but maybe only a subset. Users can be granted read or write rights to specific nodes. In the same way users can get clearance to invoke specific methods, each node or method must have lists of what each user can and cannot do. Different models exist to accomplish this task. No matter how it is done the basic task is the same. First a user identifies him/her self to the system. When the user tries to access a method or some data, it will be allowed only if the user is allowed to perform the operation. Different access control patterns can have different levels of access, however no matter how the check is performed; the system must be able to complete the task from a user credential

2.2.6 Reliability

In enterprise architecture, businesses must be assured that the messages sent actually do arrive at their intended destination. Without this assurance businesses would not be able to use web services for industrial-strength business applications and for mission-critical operations like control of a WPP. Business-to-business transactions or real-time enterprise integrations would not be possible. Web services send messages over an unreliable network connection, using different transport protocols. Therefore it is essential that the reliable message layer is defined at a level higher than the underlying transport protocols. This ensures that the reliability can be truly transport independent.

4- Specific Communication Service Mapping (this is a part of the IEC 61850-8-1 standard)

There exist several ways to define reliable messages, but the most basic level of reliable messaging simply refers to the ability of a sender to deliver a message once and only once to the intended receiver. This can be defined as follows: [PY03]

1. Support carrying message traffic reliably in support of business processes whose lifetimes commonly exceed the up times of the components on which these processes are realized.

2. Support quality-of-service assertions such as:

    1. Each message sent be received exactly once (once and only once), at most once, at least once, and so on.

    2. Messages be received in the same order in which they were sent

    3. Failure to deliver a message be made known to both the sender and receiver

3. Accommodate mobility of a reliable business process to different channels or physical machines.
4. Support message transfer via intermediaries.
5. Leverage the SOAP extensibility mechanism to achieve reliable messaging.
6. Enable reliable messaging bindings to a variety of underlying reliable and unreliable transport protocols together with the Message Routing Protocol.
7. Compose with other protocols to support security and other message delivery services.

When dealing with WPPs, monitoring and controlling it is very important that the messages sent do arrive at their destinations. If the WPPs are going to be controlled remotely, the messages must also arrive at their destinations. For this reason the system must have reliable messaging properties.

2.3 Proposal for system architecture

To make the system generic, different elements in the system must be isolated. Figure 7 below shows a proposal for the system architecture.

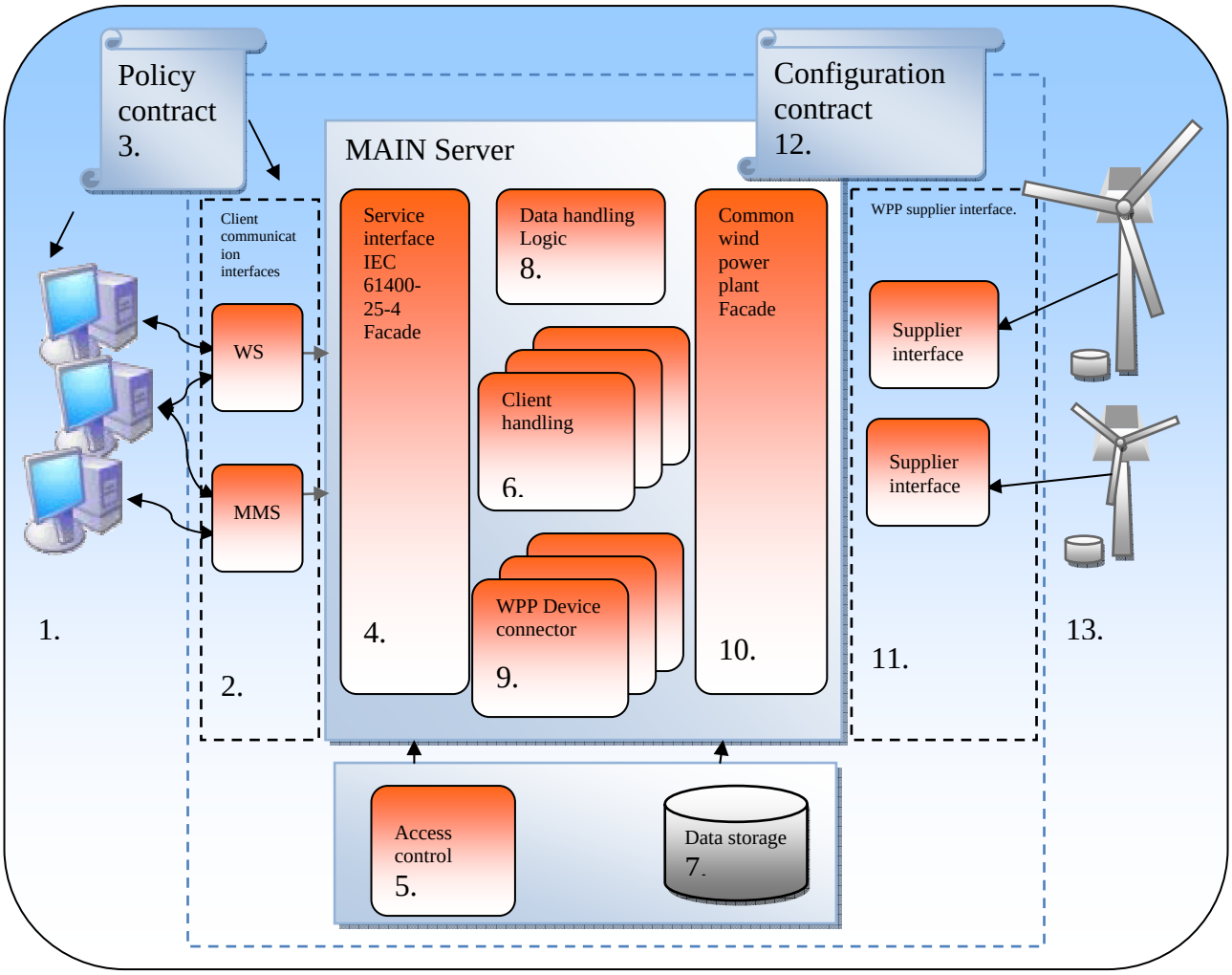


Figure 7
Proposal for system architecture

1. **Client** The client consumes the service. Clients can be web clients, applications etc. (The client is outside the scope, and should simply consume the service through the WSDL file)

2. **Client communication interface.** The client can use different methods to conduct the communication with the service. In this thesis the web services are the preferred solution, but the core system must be ready for other methods like MMS etc. The interface is defined in IEC 61400-25-4

3. **Policy files** define communication specific information for the communication. This include security information, use of transport protocol etc. By using policy file, what (data) and how (binding) things are communicated are completely separated. This ensuring an easy changes in how data is transported between the service and the client

4. **Service interface.** A common interface for acquiring the services of the system. Common interface assures that the service can always be reached through the client communication interface as long as it obligates itself to use the interface defined in IEC61400-25-4.

5. **Access control.** Checks the client's right to access a specific method or dataset. This should be a separate module in order have different access control modules.

6. **Client handler.** When communicating with the service the client can ask for specific data. The client handler handles the connection between different clients and the service. The service must at all time assure itself that all its clients are online. The client might only need information if an alert occurs, or a difference in data is detected. This goes against the common client server model, and removes the possibility of having a web client. The information exchange model defines how this is done in IEC61400-25-3

7. **Data storage**. Stores the data for shorter or longer time.

8. **Data handler logic**. Keeps track of data, and the logic to run the service generally.

9. **WPP Device Connector**. Different WPP suppliers produce different data. The device connector gathers the necessary data from each WPP. The information model in IEC 61400-2 defines this

10. **Common supplier interface.** There is no generally approved standard for how the data must be supplied by the WPP, but by defining one, supplier modules are easier to replace in the system.

11. **Supplier interface**. When connecting a WPP to the service, the supplier has to create an interface file. The file has knowledge of the supplied nodes that a WPP can expose, and how often pooling is necessary for the WPP.

12. **Configuration contract**. The contract must contain information of the data each WPP can deliver. Also the contract defines how often the data is to be pooled. This must be done in compliance to the IEC 61400-25-2 information model

.

13. **WPP**. Each supplier exposes its capabilities to get and set data through their own interface.

2.4 Summary

This chapter has provided a description of the problem domain and the specifications to the framework which is being implemented. The modeling concepts of the information model have been presented which is the very base of the system.

The analysis of the challenges in designing and implementing the system has been made and a set of proposals have been discussed which addresses the problems identified.

The data processing and data storage have been discussed in detail ending up with a solution which addresses an important problem on how to make the data available to services. The solution was seen as the most appealing to the problems concerning the processing of data.

The resulting overall strategy will be able to solve all the major problems presented. An implementation based on the proposed architecture will result in a stable and secure system being able to service multiple clients with access to the hosted devices.

In the next chapter, a technical description of how to solve the challenges with existing technologies can be done. A detailed description of the use of Service Oriented Architecture will be presented together with the important design and implementation properties.

Chapter 3

SOA AND OTHER TECHNOLOGIES

In the last chapter requirements were identified for what the system must do. Now it must be established what is needed in order to realize this in a language and platform independent way. The standard is centered on the use of web services.

In this chapter introduction to different standards and technologies will be presented. This will lead to a greater understanding of the choices made in the design, and what is gained form those choices.

In this chapter the following topics are covered:

- Service Oriented Architecture (SOA) is defined in a general way, followed by a discussion in the context of the prototype. From this discussion it is concluded that the benefit of using SOA for the prototype is greater than the drawbacks, and for that reason SOA is chosen as a development strategy.
- A description of general terms used for SOA to provide essential security functionalities for a web service.
- A discussion of web services and how Microsoft addresses them and their various possibilities related to different design approaches are presented.

Much of the described solutions can be used on several platforms and languages, but since MS C# .NET is chosen as the language and platform for the prototype to be used later in this chapter, the main focus will be on the abilities that this choice provides.

In the end a Microsoft .NET and C# are picked as platform and language respectively. SOA is chosen as the development strategy.

3.1 Introduction to Service Oriented Architecture, and why to use it the IEC 61400-25 prototype.

When talking about SOA, one usually refers to the field of software, but the idea is better explained in a real world scenario. A service can be defined in several ways. According to dictionary.com, a service is defined as follows:

"*The performance of work or duties for a superior or as a servant*"

Or

”*an act of help or assistance*”

A service can be defined as a subcontractor, performing a sudden job on the initiative of another party who needs the job done. Architecture is defined in several ways, and can just as a service be defined in a general or a more software specific way. A crude, but still accurate way to describe it is done by Fowler [Fow02]:

"*'Architecture' is a term that lots of people try to define, with little agreement. There are two common elements: One is the highest-level breakdown of a system into its parts; the other, decisions that are hard to change.*"

A more software related definition would be [BCK03]

"*The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. Externally visible” properties refer to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on.*"

Software architecture defines all the pieces a system is comprised of. It defines how the different parts are related to each other, and how these parts are connected together. Having a software architecture, all questions and considerations about communication and component interfaces should be established, and this should be done at the earliest stage possible.

If the system doesn't manage to make the individual components as independent as possible of all other components, the system is said to be tightly coupled. Having a tightly coupled architecture reduces the flexibility of the system, and as a result limits its means for reuse.

In a Service Orientated Architecture the system is comprised of several services. Each service supplier exposes one or more services (methods) to a network, ready to be invoked. The system invokes the service on demand, and the connections between the components and the client are done through technology neutral messages defined in a standard way. This results in a more flexible and loosely coupled system, in contrast to traditional system architectures. [MPP2003].

Each service can have different resources and methods associated with it, and can have different ways for realizing the communication. One of the must used once are web services. Figure 8 below shows a client using two different web services. In this case each service is illustrated as a WPP because the goal of the IEC 61400-25 is to isolate them as an autonomous service.

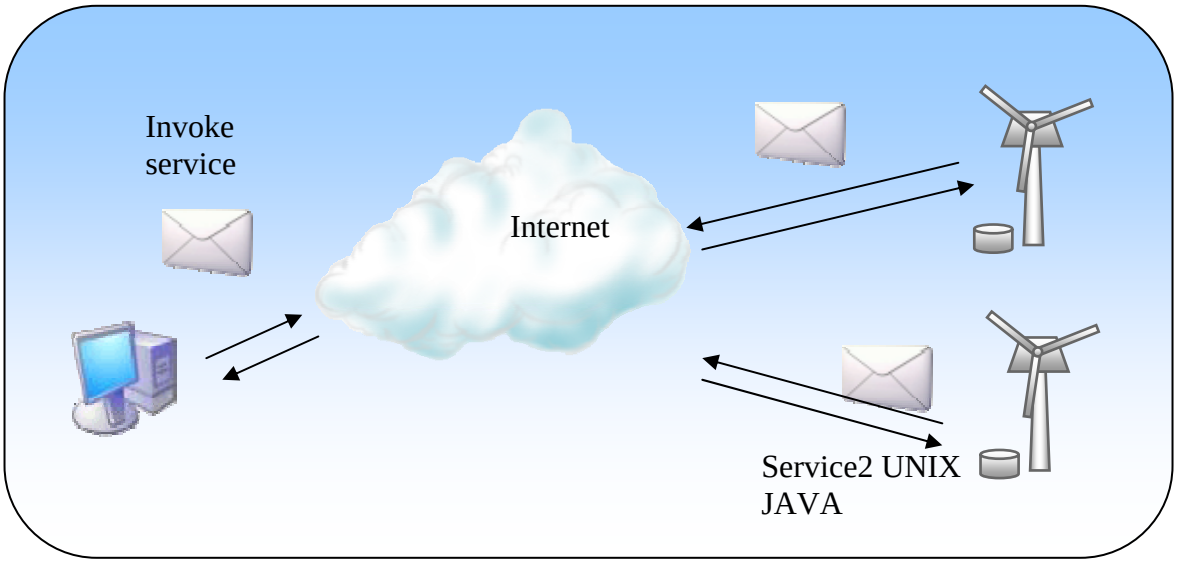


Figure 8
Service oriented architecture. Each service is running on its own platform using its own language. This is possible because of the common agreement of message format.

For many years the most used approach to solving a software problem was through object-oriented thought processes. Here the communication was done by sending language-specific objects within the system. This made a tightly coupled system, because the data could not be shared between languages or platforms. Having a SOA, the external communication is done sending messages in a neutral format, rather than passing native objects around. This implies that a system can still be designed using well-known and tested object oriented design ideas internally, as long as the external communication is done by messages having a standard format. SOA is ideal for cross-platform sharing, because of the looser binding to specific technologies. In SOA the business logic has the possibility of being distributed onto different computers in a network, because a computer in a SOA is a node that exposes resources for other nodes in the network to use. For the end user though, it must be completely transparent if a method is running locally or remotely.

Messages in a service should be non-vendor specific, making the service available to anyone who has the rights to use it. For several years, attempts have been made to design SOA based systems, some of which include java RPC[5] and DCOM[6]. Both of these are tightly coupled to a specific programming language or platform, which of course did limit the use for everybody in the field. But other and better attempts do provide SOA a

5- Remote procedure call for java, it is build on top of XML-RPC. The message is a HTTP-POST request with an XML body. It can be compared to a simple SOAP Message, but it lacks the possibility of defining a contract for the services like the WSDL provided for web service operations.
6- Distributed Component Object Model. This is an extension of the Component Object Model (COM). DCOM allows components to communicate across network boundaries. For most of the time the inter-process communication in COM is done within the boundaries of the same machine. DCOM however creates a transparent interface for COM, and uses the RPC to send and receive information between COM components on a network. DCOM is restricted to Microsoft platforms.

greater chance of becoming really big. According to Ziff Davis [IBM04] the future for SOA is bright. He wrote:

"During the next few years, service-oriented architecture will likely expand to all corners of the corporate universe: from auto manufacturers to pharmaceutical firms; from electronics companies to consumer goods."

This, Davis continues, is due to the fact that the technology has achieved widespread acceptance from all major players in the field, and the acceptance of the key standards. This of course is closely related to the use of web services.

Web services utilize a plethora of different commonly agreed XML protocols. This assures that any system supporting XML can invoke these services. Web services do not guarantee good and reusable software components in it self. Accordingly, a few guidelines should be followed when designing web services.
These guidelines address the following issues:
b

- Loose coupling
- Autonomous
- Share contracts
- Share policies

A service is loosely coupled and has explicit boundaries.

The service should be loosely coupled. There should not exist any or only very little platform or data specific information in a call. This implies that the service should keep its own data records, and any new information must be supplied by the caller. The logic of the service should be completely separated from the external interface, and therefore no prior knowledge should be required in order to utilize the service. In order for the client to use the service it must be transparent how the connection is achieved. Whether the service is running locally or on a remote machine, no adaptation should be necessary for the client. This does provide the system with a great amount of flexibility, but the very different environments in which the services can operate do add a great aspect of non-guarantied service performance. Depending on the network speed, security aspects, or other choices, the system might perform with great variance. As a designer, it is essential to keep the 'cost' (loss in performance) in mind when designing the finished system, and aim at designing according to a worst-case scenario.

Main points

- Transparent access
- Contact to other services can be considered as crossing a boundary, crossing this boundary has a non-guarantied price and consequence.

A service must be autonomous.

Services should be autonomous. If one service in the system fails, it does not imply that other services in the system will also necessarily fail. No central controlling entity must be used. Deployment, maintenance and service operations must be done locally. Each service still have the opportunity to use other services to perform a task, however it is still the job of the initial service itself to ensure that any failure, domestic as well as foreign, will not cause the system to crash, and any waiting clients will get a response, even if the response is that the task could not be completed. For standard exceptions this can be a trivial task but in order to communicate state information between client and service, they must be defined. The IEC 61400-25 standard does not provide that many standard state messages about the services, however extending the standard with more state information could provide a good way to handle state information in a common way.

A call to a service should not be dependent upon any prior information to complete its task. The information should always be supplied by a client for a service to complete its task. By having each service completely independent of each other, updates can be performed upon a single service without losing the functionality of the others.[7]

Main points

- The service is self controlled.
- The service is responsible for all communication to and from the service.

A service shares formal contracts.

Types used in the service should not depend upon specific languages or platforms. On of the most valuable aspects about SOA is the fact that service and clients can share contracts about how and what is sent between them. Formal contract includes:

- *Data contract.* Data types used by the service.
- *Service contract.* Signatures for the methods the service exposes.
- *Message contracts.* Contracts about how the messages are passed between them.

By designing the messages first (contract first), both the service and the client are forced to have the same data types, signatures etc. The contracts are defined in XML ensuring that different platforms and languages can read and understand them.

Changes to a contract have influence upon both service and all the clients using the service. For this reason contracts should be subject to changes as little as possible. Necessary changes in a service should include the definition of the versioning of the service, and when it is possible to define changes as new service instead of changing

7- Stateless services are not always a practical solution. In building a complex system, access is often required to use a service. It's possible to exchange credentials with each call, but it doesn't utilize the resources very well. In most cases you would simply place a token on the service side containing the credentials information of the client

existing ones. This does create a legacy system, however if many clients were depending upon a service, it would probably not be wise to force all clients to implement the changes. In appendix C a brief discussion about changing contracts is presented.

Main points

- Service and client share contracts
- Changes to an agreed upon contract should be considered deeply before enforcing the change.

Transport and compatibility of the service is based upon policies.

Contracts describe the information that the service and the client must agree upon, however it is also possible to have contracts about other aspects in the communication, such as security information, or the transport protocol the service can use. These aspects have something to do with the quality of service that the system provides. That is overall performance of the system. Service and client must agree upon the policy in order to work together. Policies deal with aspects of the communication where it is not guarantied that any client can meet the requirements. Custom protocols can be placed upon the transport, or different approaches from different vendors might restrict the use of the service to a particular platform or language. Some common policies for the communication do exist and can be used across platforms. The policies are “wrapped around” the communication. This ensures that they can be changed without changing the underlying service. Changes in the communication can be done simply by changing or adding a policy to a service. A service can have many policies for different kinds of communication. Two systems supporting the same kind of profile and contracts can always communicate.

Main points

- Communication requirements can be defined as policy.
- Changing policy alters the requirements for the client on how communication is performed.

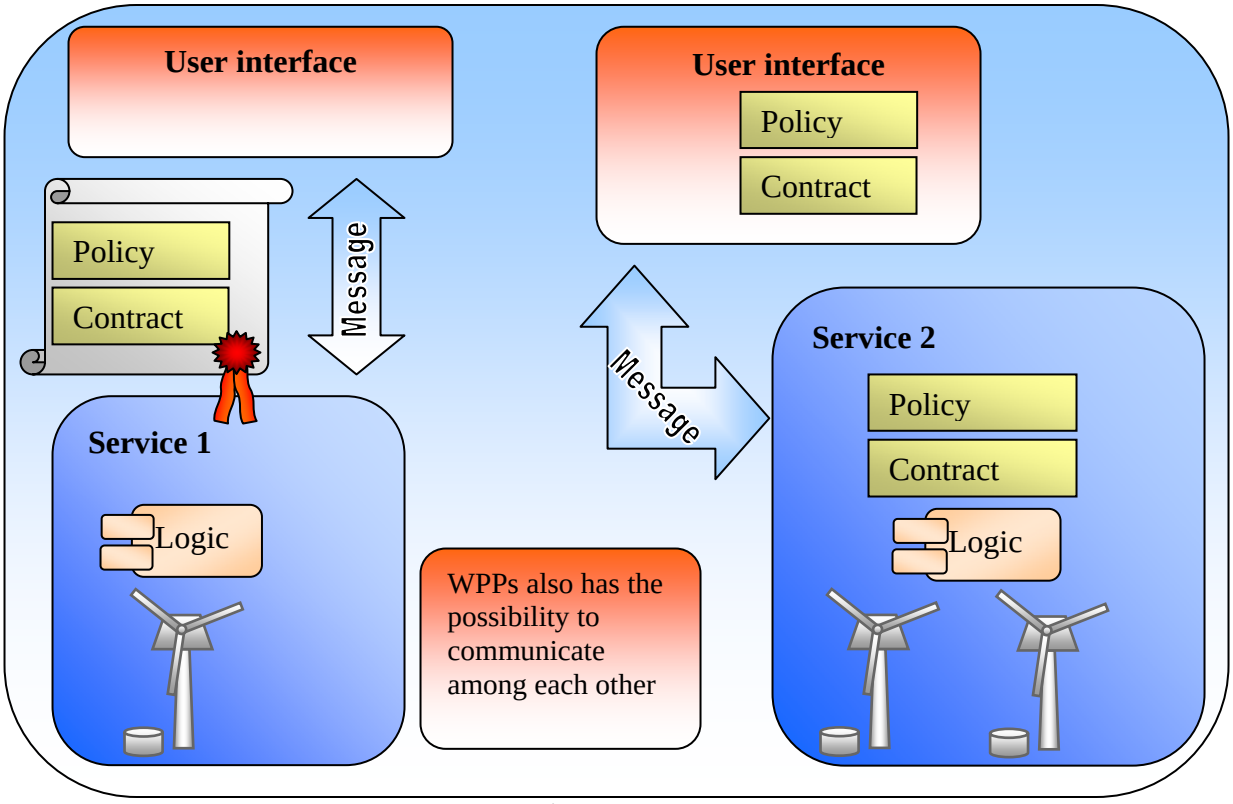


Figure 9
SOA

Figure 9 shows how policies and contracts can be used. Notice that they can be shared at runtime where the information is read whenever the client tries to connect to the service, or alternatively at design time where the contract and profile is built into both service and client. Especially the feature of reading the policy at runtime is valuable. This enables service to be changed without having to rebuild the system. For instance, the transport protocol can be charged, or security can be added simply by changing a file holding the policy for a service.

When implementing the system described in the previous sections, a number of different technologies must be used. In the glossary a brief description of each of the key technologies for a web service is described.

3.1.1 What advantages are there by using SOA in implementing the IEC61400-25 prototype?

SOA can be used in many different environments. But in the case for this prototype the amount of different vendors and individual requirements in the use of the system places the scope of prototype as an enterprise. The following discussion therefore will only look at SOA for WPP as enterprise architecture.

The purpose for designing a standard for controlling and monitoring WPPs is to have a common way of performing certain actions. As the designers very well know, it is hard to get a group of people with different interests to agree upon a common standard unless it is flexible enough to suit their individual needs. This not only includes *what* is communicated, but also *how*. IEC61400-25 uses web services as one of its communication mapping profiles, defined in IEC 61400-25-4. Using web services ensures that different clients and environments can be used. Each vendor can develop the service or client in there favorite programming language and host system. The system will still not be tightly bound, as long as only standard methods and data types defined in the contract is used. Each vendor is not dictated to use a specific language or platform.

Other benefits include:

- **Updating:** Having a loosely coupled system comprised of services rather than a tightly coupled system, provides the benefit of updating only subparts of the system. This can be done without a system-wide shut down. Only the involved services are turned off. If the system has several possible places to get a service performed, the system might not even look any different to the end user. The internal business logic can be optimized endlessly, and as long as the interface remains the same, there is no need for rebuilding or taking the whole system offline.

  It is intended that a service control one or a couple of WPPs. The effect of updating in one service will not affect other WPPs.

- **Maintenance:** In a system comprised of several fat clients[8], changes to the system would have to be introduced to every single machine in the system. By using rich clients[9] the functionality only has to be changed at one place. If each WPP has its own web service it is still necessary to deploy changes to each individual WPP. But if a service is used by several clients this would be a considerable advantage; like a logging service used by several clients.

- **Reuse:** From the use of OOP we know how reuse of logic and components in a system can reduce the development time drastically. These components are often tightly coupled to other classes, or a specific environment. The contrast to this are services where ideally there exists no binding to other components in the system. This makes the service ideal for reuse across a variety of platforms and languages. The different WPPs are not interchangeable, because they hold their own unique data, however the *logic* used on each WPP can be reused on WPP with the same configuration.

8 Fat client are systems where the main processes of a system are often relies upon a single backend system, maybe with a database connection.

9 Rich client is a system where the client can use different services, and keeps track of its state information.

- **Scalability and availability** One of the major problems facing a system is frequently changing demands. The number of service calls changes, and the hardware cannot handle the requirements anymore. To accommodate this, a good system should be easy to up- and down-scale to meet demands. One strategy might be to distribute services to different locations, having more machines doing the tasks of the system, thereby balancing the load. In traditional operating systems, it might not be easy to distribute the system. At least it would require that some kind of redesign be made. This of course could be a costly affaire. By using services, each service could simply be deployed on several machines. The additionally offered services then just had to be registered in a Universal Description Discovery and Integration (UDDI)[10] directory. It would be possible to split the service associated to a WPP into smaller parts, where each service only is in charge of a subpart of the methods that the WPP exposes. How ever most of the methods rely on having access to current data from one specific WPP. Here it would not make sense to split the service. For logging and reporting it might be plausible, if one service could not handle all the clients.

- **Cheaper to change.** If the system is designed in a tightly coupled manner, changes might be very costly. Change of a supplier might not be as easy as in theory. A lot of work might be involved in either converting the existing system to the new supplier or some of the systems dependent objects might have to be redesigned and implemented again. From a business point of view, the two major reasons for not introducing new and possible better solutions to a system are the time that it would take to implement the change or alternatively that the tasks are simply demands to many resources to. The new functionality might be beneficial to both customers and the firm in general, but it is not always economically feasible to enforce the change. The total cost of the conversion must be smaller than the short or long term cost of the change, to save money. Having a system comprised of completely independent, interchangeable objects would resolve in far cheaper restructuring of a given system. This opens the possibility of testing new and possible better solutions or services to a system, making the overall value higher. [KBS04] With the amount of players in the WPP field, using the standard opens up for the possibility that the proportion of shelf services becoming available is rising. This could be a far cheaper solution when the cost of designing and testing a component from scratch would cost.

If a SOA system is designed in the right way, it assures that the system be agile, where changes and maintenance can be done in an easy and cheap way.
It is very important for the system to be flexible, where changes can be administrated in a cheap and easy way. If the system is designed in a way where functionality is too tightly coupled, the cost of making a change might simply be too expensive to perform.

10 UDDI is a platform-independent XML-based registry for services to list themselves. UDDI is an open industry standard, sponsored by OASIS. UDDI is a directory of services there can be used by others. It can be compared to a phonebook advertising the existence of a service. UDDI is part of the Web Service Interoperability (WS-I) standard and is considered a corner stone in the web services infrastructure.

The most significant reason to use SOA for the IEC61400-25 prototype is the possibility for each vendor to choose their own platform and language for their service or client, and at the same time ensuring that other vendors can still communicate with them. This makes it a lot easier as an owner to maintain the system, and data supplied by the WPP can be compared directly. More players in the field will also benefit the cost of developing new monitoring system, due to a more competitive marked.

3.1.2 What are the disadvantages of using SOA for the prototype?

- **More costly to develop:** Having the extra layer added to the communication does make SOA a more expensive solution in the first place. In a smaller system where the specifications are not subject to change, or there is no desire for reuse, SOA should not be used. It is important to remember, though, that the longtime cost of having a SOA system could be far lower when compared to the cost of maintenance and update of a non-SOA system.

- **Speed critical application:** Having all communication done over SOAP does add an overhead on the execution of an application. The data amount is far greater than by using regular methods. This implies that limited bandwidth or a very time critical system might need another solution. Also depending on how to deploy web services, today one might require basic security requirements to be fulfilled. Adding this extra security layer also adds an extra overhead. The parsing of the XML and the time used to assure security ads to the system overhead. (It is always possible to upscale the system to handle the extra bandwidth or processing time needed. This would make the system more expensive though). Some of the WPP does reside in locations where communication relies on a modem. In this case, the data amount is the critical resource compared to the speed of the connection. Further, the system does have something to gain from compressing the data being sent. This will be disused in later sections. The prototype for the IEC standard is prone to producing a system that relies heavily on high-volume data transmittal. At the same time storage and bandwidth-critical hardware demands that the design addresses these problems one way or the other. In the initial proposal of the standard it is not intended that the WPP should be able to be controlled on a time critical level, and a such this point is not that important, however if WPPs are suppose to be an variable node in the overall power grid, it is essential that it is possible to regulate it real-time. In the future this will properly be addressed through various research projects.

- **Offline scenario.** SOA depends deeply upon the availability of a network connection. In the case where these services are unreachable, the system is useless. To make a usable system even in this scenario, additional measurements must be taken to assure a working system. This of course adds to the development time and the total cost. From time to time, a WPP might loose its connection to the main server. In this case, the data must be delivered at a later time, once the connection is reestablished if the data still has a value to the client. If the WPP is being controlled,

and it is solely depending upon the information it gets from a central control unit, some kind of redundant communication path must be established. As a result of this, the development price goes up.

- **Vendor is no longer in charge of what to implement** .Each vendor is depending upon a standard to define data classes and information that can be send. It is possible to define customary messages, but the usage will remove the vendor independent aspect of the communication, not completely, but it could force a client to use specific software in order to utilize special services.(This is solely seen from the vendors point of view where they would like to keep product control of their own WPP. From the perspective of the client having a common standard will be a great thing.)

All of the described performance disadvantages can be solved by adding more hardware and design time into the project. On the bottom line this means that SOA will be more expensive to develop up front, but still the longtime cost might be lower. Also it is important to keep in mind that the relative extra cost is rather small compared to the cost of the WPP itself, or the cost of going offline with the WPP.

3.1.3 Choice of SOA as a development strategy for the server prototype

The major communication mapping defined in IEC 61400-25 is based on web services. By using web services, the necessary platform and language independency requirements can be met. Numerous clients are going to communicate with each WPP, and it is essential that effective communication can be achieved. By having a service or an entry point for each WPP, new WPPs can be added to the overall system (WPP farm) simply by adding it to the UDDI. This will easily notify the client of the new WPP. The cost of implementing a standard of this size should compensate for the extra people adopting the standard. If the standard can be commonly accepted the cost of developing new components can be reduced drastically due to large number of players sharing the cost of developing new and better components.

Unfortunately, maintenance of the service must be done on an individual basis because each service is not interchangeable with any other service due to the unique data each WPP holds. This goes for the update and security aspects of the WPP. A protocol for managing the WPPs in a centralized manner can be made, but there is currently no standard way of doing this.

The vendors of the WPP might have very different approaches to how the control structures of the WPPs are done. SOA can be used on many different devices. By having the distributed nature, the system resources needed for end clients can be kept at a minimum by unloading some of the work onto the fastest machines in the system. This ensures that even systems with limited resources can be used, like a PDA.

The speed requirements are not easy to analyze for the system. Depending upon the individual vendor and supplier, the amount of data (how many nodes are monitored), and the interval in which the data is gathered varies a great deal. For this reason, it is not possible to say anything specific about the implication that the use of web service will have on the system. A great amount of data must be collected, but how to optimize it surely depends upon the particular requirements of the end user. Different measures can be taken to address a particular problem for instance adding more computer power, or compression of data. Given the nature of web service though, up scaling can be done in an easy way. Controlling the power systems is big business with a great budget. Adding off the shelf hardware for an up scaling would be considered a minor expense compared to the total budget. Therefore this should not be seen as a reason for not using web service in this case.

The biggest negative side to using SOA is the offline scenario, which is a connection interruption between then client and the WPP. It is vital to have a stable and reliable connection to the WPP. No suggestion has been directly made on how to avoid this problem. However, using intercommunication between WPPs, the communication path could change so that a virtual identical connection could be established through a neighboring WPP to connect to the target WPP. This might not be the best solution though. Most likely, the connection will be lost to the wind farm in general, and not to an individual WPP. For this reason, it would be good to have several ways to transfer data, for example a wired internet or a satellite connection. In any case Data must not be lost due to failure. This calls for the system to buffer the available data on the WPP until it is fetched by the client (a client seen from the WPP). The WPP must at all time be independent form control information, however if it is preferable always to have control contact, and the WPP can not wait until it gets online, the alternative communication form is a must.

The use of SOA for implementing the standard is a wise choice. First of all because the most vital reason to create the standard in the first place is vendor independent communication. This also includes the freedom for a vendor to choose the preferred environment and languages for the implementation. The higher cost in development is considered a minor expense compared to the benefits the use of SOA provides to the system.

3.2 State of the art: SOA

Web service and SOA have been around for a long time. For SOA to be a success several aspects of the communication, concerning everything from security to compression, must be addressed in a standardized fashion suitable to most people. A great deal of work has been put into creating industry standards for all kinds of aspects of web services. The following section will describe some relevant aspect need for the prototype.

3.2.1 WS-I

WS-I is an open industry organization which gathers common web service scenarios, and defines a standard for solving common problems.
They promote interoperability among services created in different languages and on different platforms. Many different aspects are addressed by WS-I, chiefly among them are guidelines for use, security, requirements, et cetera.

WS-I has defined the basic profile. Stating that a service conforms to this profile is not a guarantee of interoperability, but a common way of addressing problems. The profile simply ties the different standards and their versions together. To conform to a profile, only the standards accepted in that profile can be used. Having different profiles also ensures that a service can be backward compatible, and still allowing new features to be added. A great number of profiles have been defined, each of them addressing new problems, located in different places in the communication stack.

When dealing with web services it is possible through profiles to configure traffic. This is a great advantage because different issues can be addressed without having to change the code as long as the client and the service share a common profile. [WSI]

The different policies is all called something starting with “WS-“ followed by a word describing what the policy is addressing. The profiles are known under the common name *WS-**. Some of the most important profiles are described next.

3.2.2 WS-Policy

This is a standard for defining a policy for a service. The policy describes several things such as security quality of the service et cetera. The policy defines how the binding is done for the service.

A WSDL file can be extended with policies defined using WS-policy standard. This enables the consumer of a service to also get the policy necessary for using the service. Another way is to have a policy file for the service. This file can define different aspects needed in communication with the service. For this thesis the service and the profile has been separated in order to keep configuration easy.

3.2.3 WS-Metadata Exchange (WS-MEX)

Using metadata exchange enables a service or a client to get WSDL information dynamically. The metadata exchange is used for getting information about a service whenever possible. WS-MEX is very important if the client want to create a proxy for the communication. This exposes the WSDL to the client, enabling him to create the proxy. It also enables information about policy data schema, in turn enabling versioning of the service. Later policies are described. Changing them enables a service to change operational conditions at runtime.

3.2.4 WS-ReliableMessaging

Reliable messaging or reliable sessions guarantee that any message send between two endpoints is delivered exactly-once and in the right order. It is done in a transport neutral way, and is not restricted to specific protocols. At the same time it is not restricted to a point to point connection, but can be used across different transport connections. Reliable sessions are part of some of the standard bindings, and can be included in a custom binding. The system has been configured to use reliable sessions. Communication do not have to be monitored managed in the implementation of the IEC server, but it can be completely separated from the system it self.

3.2.5 Data compression

Even though SOAP is a great human readable protocol it does add a lot of overhead to a system that can get precisely the same information out of binary data. Today the network connections are getting faster and more efficient. For this reason, one might neglect to look at the amount of data transported over a connection, and what implications it might have on the overall performance of the system. Several ways exist to reduce the overall data amount transported. If the web service is to serve many individual clients, compression would have a tremendous impact upon performance since performance is dependent upon compression and serialization speed.

A test conducted using zLib compression can reduce the message size about 50 to 70 percent, but at the same time it does add computing time to a roundtrip call, hence reducing the number of calls handled. ZLib is a text compression algorithm widely used in HTTP servers.

If the system is to serve many individual clients, compression could slow the system down resulting in low performance. However, if the service is servicing a single WPP connected to limited bandwidth connection, compression would limit the data needed to be transferred, but at the cost of the central server paying the added workload. [CKRS2003] Zlib is a Zip based compression. Zip is an open standard, and for that reason it does not limit the use of it to a specific platform. Unfortunately there is not defined a standard like WS-Compression for the time being. But compression is part of

the HTTP 1.1 protocol. In this prototype a custom filter has been used doing the compression.

Another data format is Message Transmission Optimization Mechanism (MTOM); this is a W3C controlled format placing binary data direct inside the SOAP as MIME objects. MTOM compress some of the XML data, but not all of it. MTOM is mainly for compressing binary data like executable files or image files, for this reason it does not prove a good solution for simple XML/text transfer as test shows later on I the case studies

3.2.6 Code first or Contract first

When working with web services, service and a client share different documents defining a contract for the service WSDL and policy files. The program code and these files must always reflect the same idea. There are two main ways in which this can be achieved. “Code First” and “Contract First”.

In the code first approach the classes is coded in a conventional fashion. Some attributes are added to the code, telling the compiler to interpret it as web service related code. When the code is compiled, a resulting WSDL file is generated. The focus of the approach lies in the *methods* and not the *messages* which are to be exchanged. This usually has a positive effect on the development speed. The downside to this is that it is up to the compiler vendor to specify how the data is exchanged. This could result in messages that are depending upon a specific language or platform, as described later on.

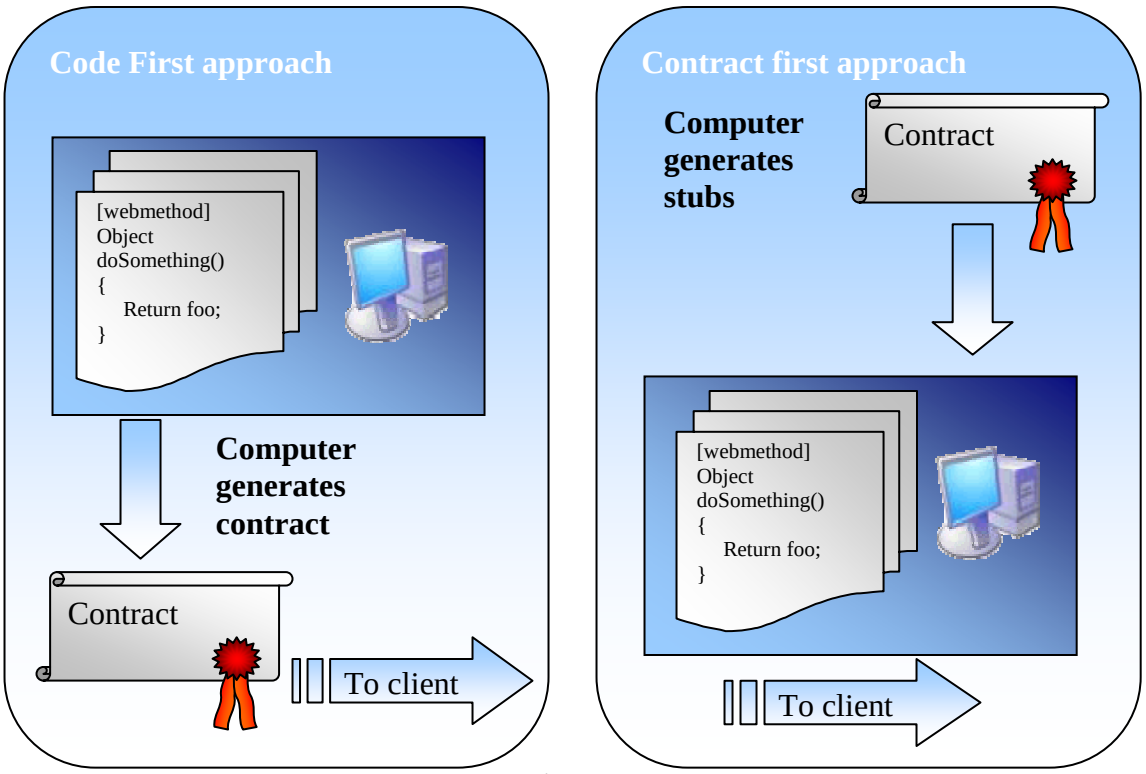


Figure 10
Contract first and code first approach

In a Contract First approach you write the WSDL/XSD file from scratch. From the WSDL file or the XSD file you can then get the basic code generated for your preferred programming language. The tool generates a stub that can be used for calling and receiving data from the methods. By using a Contract First approach you minimize the dependency upon platform dependent types. (This of cause is dependent upon the vendor performing the XSD conversion in a manner conforming to standard data types.)

Giving the whole idea of web services you should in an ideal world care less if you use one or the other. Everything should work just as fine, regardless of language or platform. However, this is not always the case. A developer writing a service should consider if other platforms and programming languages are supposed to consume the service. A system designer has the possibility to specify via attributes, more specifically how the serialization must be done. If no attributes are associated with the code, the serialization of the data is done implicitly, meaning that the compiler may decide how the serialization is done. This will be reflected in the SOAP message, and can lead to problems if dealing with language specific types without serialization attributes specified.

When choosing an approach, one must decide what the most valued attribute of the system is; the messages and interoperability, *or* methods and productivity. From this choice either a Code First or a Contract First approach can be chosen.

IEC 61400-25-4 defines the WSDL and SOAP messages which are to be used. This implies that contract first must be used. On the other hand, the defined messages take

for granted that no additional information is passed inside the methods. When security is added, or some "must support" things are added, the SOAP message will contain more data than the one described in IEC61400-25-4. The WSDL file will reflect the choices made. As a result of this, it is not possible to directly generate a stub for the web service from the supplied WSDL file. And in this way consume a service using the same WSDL. The structures, signatures and data types defined must be used, but how the precise SOAP message looks depends on the profile used[11]. A proxy must be generated from the generated WSDL created by the service.

Regarding the choice between Contract First or Code First approaches, both have their advantages and disadvantages, but for the time being you simply can not just use one or the other, you need to be aware of what happens all the way through the message stack.

3.3 Choose of platform and language for the prototype

In choosing the platform and language it is important to establish how and what the system is to be used for. The main idea of a web service is that it is supposed to be completely platform and language independent. No matter what language or platform is chosen as the service language, the client can be in any other language. For the time being, there are some differences in how the individual language vendors implements and uses the standards. Their number of standard features described above is also different. The fact that a language support web service doesn't guarantee that it supports things like the WS-*. To make sure that any language is working with the prototype will require some research. For the purpose of this thesis, only research of inter-compatibility has been done for a couple of languages.

The two most used enterprise level platforms for web services are for the time being Java and .Net. The nature of .Net implies that any languages used under it will be able to work together. The language uses the same objects, and for this reason has the same web service features no matter which .Net Language is chosen. .Net works with all the most common languages for windows such as C/C++, J#, C# and Visual Basic, to name but a few. All of these require the use of Windows-based systems. It is possible to find frameworks for running .NET on UNIX derived operating systems (MONO); however they are still in their early stage and not mature enough for production.

Java however is not a .Net language, but it is platform independent. In the design it must be considered what can be done to keep the possibility to have Java and a .Net Language work together. To do so it is essential that the input and return parameters of the service operation are interoperable. The underlying technologies, however, do not have

[11] In theory a client and a service should be able to take the same WSDL file and generate serializing data, and be able to talk together. But for the time being this can not be guarantied, because all languages should then support a one-on relationship with SOAP data types. Restrictions are not supported very well.

direct relation between their data types. When data is sent or received as SOAP, the data is serialized to and from XML data, and conversion between native types is inevitable.

Language specific data structures can not simply be passed on, even though it might be possible to pass it to SOAP. The XSD guarantees that the send type can be reproduced at the other end, but for the data to really be compatible, the languages need to have a one-to-one relation between their data types and structures.

There are three main problems that must be considered when designing a system there are to work with both java and C#. If an array is passed across as a SOAP message, C# emits any null elements, removing them from the SOAP call, while Java does pass a null element. This would in the best case only influence the size of the array, but it also might result in missing elements if position in an array is important or a null value has a meaning in the context of the implementation. The standard do have arrays, but it does not call for position based information to be passed on as a mandatory feature.

Primitive data types can also cause problems. For the data types to be safe, a one-to-one translation between that language and the XSD used to transport the data through SOAP must exist. Java does not support unsigned numerical types, while C# does. Either the conflicting data types should not be used or a wrapper class is needed. There also exists different ways of representing precision types in different languages. This could result in loss of accuracy of a data type.

The last thing is native types. Both java and c# contain an *ArrayList* class. Even though the names of these classes are alike, the way they represent data is not alike. Using *ArrayList* will not flag an error, because all the compiler needs is information about how the class is to be serialized. In C#, *ArrayList* implements *ISerilizable*, and can therefore be serialized. To avoid conflicts, a program should only use simple data types, but still the programmer must be aware on how conversions are done. [WY05]

A more general problem is that even though the standard spends a lot of time in defining new data types, it is still up to both client and server to check these restrictions. When send data is send at ASCII characters, and as of such the data should be checked for validity. When generating a proxy it is very limited what kinds of checks are automatically generated.

In the future, we must assume that a common standard for how things can be serialized is developed, but until then we must remember to take these incompatibilities into account when designing our services.

3.3.1 Comparing .NET to Java for the development platform

C# is part of the Microsoft .NET and as such runs using the latter Common Language Runtime (CLR). This is like the Java virtual machine, an interpreter running on the host machine reforming the operation of the code. Both of these runtime

environments have the advantage of "write once, run anywhere" portability for their code. Code written for such a runtime should run unchanged on any version of that runtime. Part of Java's appeal has been the availability of the JVM on numerous platforms such as Windows, UNIX and Mac etc. However, .NET comes for Windows, and as an open source project named Mono.[12] This gives Java and C# the same level of platform independency. C# and .Net however do have great advantage in the way the language handles the use of Web services. Web service in Java still involves writing a lot of the SOAP messages on the XML level, but initiatives has been taken in order to move way from this. Since the first web service implementation, Microsoft has extended the programming model several times. This has been done through their Web Service Enhanced packages (WSE). WSE wraps up and automates several of the described technologies. For instance, using WSE, End-to-End security is possible. Sometimes WSE has simplified the use of web services greatly by analyzing the most common problems programmers have been facing using web services, and placing them into common easy to use profiles.

With the advent of Microsoft's Windows Vista, a great leap has been taken towards integrating XML in many of the aspects of the system. Vista ships with the new .NET 3.0 in which the new Windows Communication Foundation (WCF) has been integrated. As the name implies, this is a new communication structure for running and connecting distributed systems. The cornerstone of WCF is web services, and WCF like WSE has advanced support concerning many of the elements described by WS-I, and Oasis. WCF also connects all of the prior distributed technologies that Microsoft used, like COM, DCOM, COM+, Enterprise Services, MSMQ, and .NET Remoting. All of these are united under a single programming model. WSE 3.0 and WCF do provide most of the same features, but WCF provides a new way of isolating different aspects of the service into more logical units. MS promises that they will keep WCF 100% backward compatible with WSE3.0 through service packs, but by moving forward the integration of older described technologies can be introduced much more smoothly. Some Java projects are being developed right now, but .Net has a great head start, and is therefore the most logical and sane choice as a development platform. This choice does not mean that Java can be forgotten as a language. If MS is the only entity implementing the standards it would limit the use of WCF until others languages support the standards. Some initiatives are for the moment implementing something similar to WSC for java. In appendix G some of these initiatives are described.

12- To some extend the MONO project do provide a cross platform environment for .NET. MONO is an open source project, whose goal is to provide a cross platform .Net development and deployment environment.
Mono is running on all major platforms like Linux, Solaris, UNIX and MAC, and like .Net it support several languages. However, benchmark tests show that the speed of MONO compared to .NET is quite slow. Performance test of MONO 1.1.3 http://www.shudo.net/jit/perf/
Current version 1.1.13 (no newer performance test has been found)

3.3.2 Windows Communication Foundation and how it address web services

The following is a description of how MS has solved the different problems identified earlier on. Only the most common ways, or the specific ones to be used in this thesis are described here.

As already described, WCF is part of Microsoft's new development framework .NET 3.0. This is a new managed code programming model for Windows. It combines the well known powers already known from .NET 2.0, but it brings four exciting new technologies to the table.

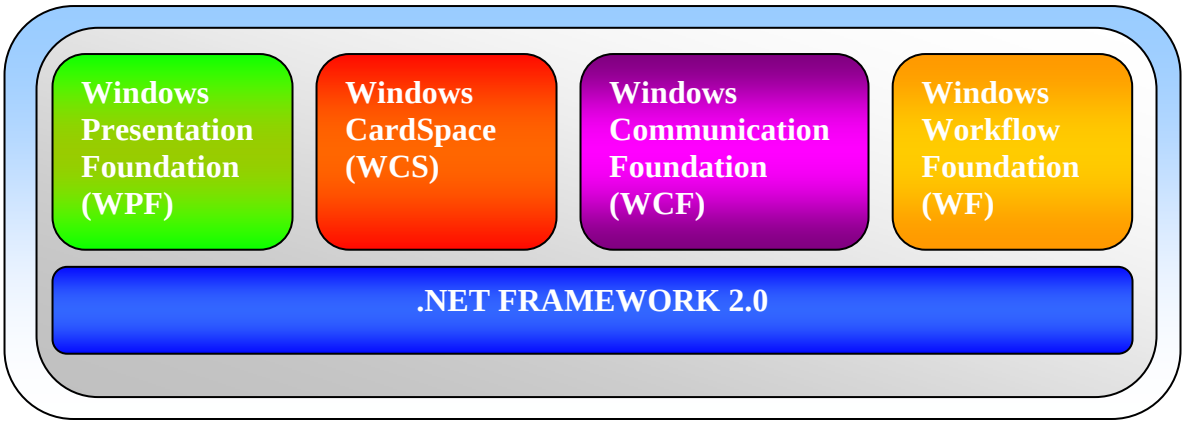


Figure 11
Microsoft .NET 3.0

Windows Presentation Foundation (WPF): This is a completely new way to interact with the user. It takes care of the Graphical User Interface (GUI). It is an XML based programming environment which enables 3D desktop programs and user controls. Multimedia context and documentation is also a vital part of the foundation, and it can harness the power of the computer's graphical adapter and thereby better utilize the capabilities of the computer. The main goal of WPF is to build applications providing a great experience for the end user.

Windows Communication Foundation (WCF): This refers to technologies connecting the system. WCF is built as a service oriented programming model. For this thesis, only WCF is important.

Windows Workflow Foundation (WF): It is the programming model engine and tools for quickly building workflow enabled applications on Windows.

Windows CardSpace: It is a component that provides the consistent user experience required by the identity meta-system. Its greatest task is to provide secure interactions with the end user.

3.3.3 Endpoint

In a distributed system, an endpoint is defined as a node where the communications start or end. Having a WCF service, one or more end points are exposed to the world. The client and the service are each defined as endpoints. The traffic route might contain some intermediate points, only passing the message on.

To use an endpoint some basic information is required. The information is a contract between the service and the client. To use the service of an endpoint, the client must conform to a commonly agreed contract. WCF has a pattern for the endpoint called ABC. This defines everything needed for the communication.

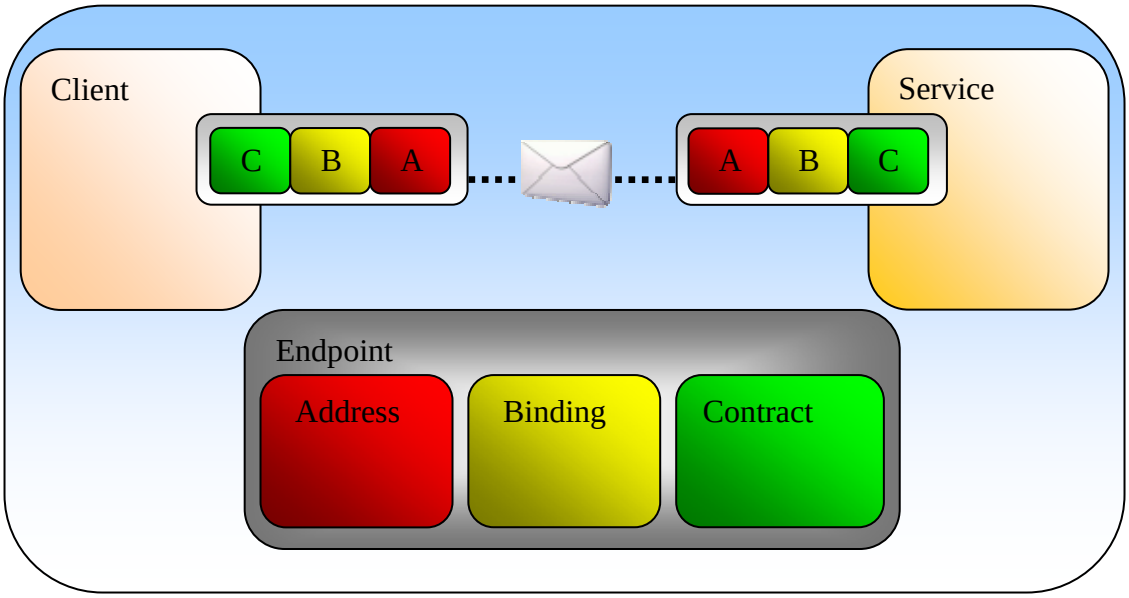


Figure 12
ABC for WCF

The unfolding of ABC:

- **Address.** The address specifies the URI that the service is located on; this can be any URI, either global or local. The address can be a server hosted service using different protocols. (HTTP, TCP, HTTPS etc.)

- **Binding** .The binding is the way the communication is performed. Binding information includes the chosen protocol (e.g. HTTP TCP) encoding (text, binary.) security requirements (transport level security like SSL, or message level security like SOAP security.) Several binding elements can be added to a service. Each element is added on top of a communication stack. And do not influence the business logic. Several standard bindings are defined such as security reliable communication and session safe communication, but there is always the freedom to add a custom binding layer if deemed necessary. WCF comes with several predefined standard bindings with different attributes. The binding has great influences on the quality/performance of the communication.

- **Contract.** It specifies what the service and the client can communicate. A service contract defines the operations supported. A data contract defines the data types exchanged. The semantic properties for the messages exchange can also be defined according to known patterns such as one-way, duplex and request/reply. Each contract type can have numeral attributes defining special behaviors for how the communications are to be conducted during communication scenarios. Each contract is split into three subcontracts defining different things about the communication.

The great strength of this pattern is that each element can be defined either in code or in a dedicated configuration file.

3.3.3.1 Binding

Different bindings contain different attributes. Below is table showing the nine standards. The list is not complete. All the listed protocols support security except the last two ones.

As the table shows, five of the nine bindings use a .Net profile. For this reason they are well suited for platform independent communication. This leaves the basic profile, the WS profile and a custom defined profile. However, the other profiles should not be forgotten, they can still provide a valuable service. Notice the possibility to use MTOM as encoding. This is a W3C standard for binary packing XML. MTOM can also be used for sending attachments with the message. This will be useful if a binary file has to be sent to a WPP. This could for instance be a file containing updates for the WPP. This will lead to smaller SOAP messages, and might be valuable in modem connections. It is also possible to use binary[13] encoding for the data. The choice of encoding will have great implications for system performance. Under profile, a discussion of this is presented.

13- For the time being binary coding is only possible between a WCF service and client.

Binding Name	Interoperability	Encoding	Transport	Datagram	Request-Reply	reliable	Duplex
BasicHttpBinding	Basic Profile	Text	HTTP, HTTPS	Yes	Yes	no	No
WSHttpBinding	WS Profile	Text, MTOM	HTTP, HTTPS	Yes	Yes	Yes*	No
WSDualHttpBinding	WS Profile	Text, MTOM	HTTP	Yes	Yes	Yes	Yes
NetTcpBinding	.Net Profile	Binary	TCP	Yes	Yes	Yes*	Yes
NetDualTcpBinding	.Net Profile	Binary	TCP	Yes	Yes	Yes	Yes
NetNamedPipeBinding	.Net Profile	Binary	Named pipe	Yes	Yes	Yes	Yes
NetMsmqBinding	.Net Profile	Binary	MSMQ	Yes	No	If transactional query are used	Yes
MsmqIntegrationBindin	MSMQ	Text	MSMQ	Yes	No	Guarantied by underlying transport	No
IntermediaryBinding	N/A	N/A	HTTP, TCP, named pipe	N/A	N/A	Yes	N/A

Figure 13
Profiles and their standard support features

3.3.3.2 Service contract

The service contract defines the service that is supported. In other words, this is the method interface that clients and service must agree upon. It also defines what is required in order to talk to a service. For instance it is possible to define that the binding must use a specific type of security.

Information about sessions, debugging information, and running condition can also be set either on method level or on interface level. In C# code, the declaration of a service contract is done through the use of attributes.

```
[ServiceContract(Session=true)]
    interface IECWCFInterface
    {
        [OperationContract(IsInitiating=true)]
        LogonReply Logon(LogonRequest inp);

        [OperationContract(IsTerminating=true)]
        LogoffReply Logoff(LogoffRequest inp);
      …
      …
         [OperationContract]
         GetLogStatusValuesReply GetLogStatusValues();
    }
```

Figure 14
Service Contract

A service contract contains two major parts:

The *ServiceContract* tag is immediately followed be an interface defining the name of the contract. Each Method in the interface that is to be exposed must have an *OperationContract* tag. Omitting this tag will not make the method available as a web service method[14].

A very useful feature is the possibility of associating something with a method call. In the service contract above Session is set to true. This enables the use of session variables in the system. With the session attribute the operation contract parameter *IsInitiating* and *IsTerminating* are both set to true in Logon and Logoff respectively. This automatically creates a session variable and terminates it when either logon or logoff is called.

A system that wants to consume a service must implement the interface defined by the service contract.

3.3.3.3 Data contract

The data contract defines how the data is transported between client and server. The data contract holds information about serialization, either as implicit types where simple data types are used, or by attributes associated with the classes defining how serialization is done. If a class implements the interface ISerializabel, the attributes can be used for serialization. As described earlier on, it is important to keep in mind what is transmitted on the line through serialization. This concerns both the name used and the type of the data. WCF comes with a new serializer '*DataContractSerializer*'. MS defines this as a Contract First system, but actually attributes the WSDL inside the code. *DataContractSerializer* can not be used in this case because it does not meet the mapping constraints defined in the WSDL (for instance the maxlength attribute). The

14- It is also possible to define the operation contract inside a class, however by having an interface, the contract and the implementation of the methods can be separated completely.

DataContractSerializer handles only a subset of XSD types. To have better control over the serialization, *XMLSerializer* must be used instead[15].

```
Using System.Xml.Serialization;
    [System.SerializableAttribute()]
    [XmlTypeAttribute(AnonymousType=true)]
    [XmlRootAttribute(
      Namespace="http://iec.ch/61400/ews/1.0/",
      IsNullable=false)]
    public class SetLCBValuesReply {

      [XmlElementAttribute(
      ElementName="AccessResult",
      Type=typeof(tSetDataValueResult),
      Order=0)]
      [XmlElementAttribute(
      "ServiceError",
      typeof(tServiceError),
      Order=0)]
        public object[] Items {
            get {…}
            set {…}
        }

        [XmlAttributeAttribute()]
        public string UUID {
            get {…}
            set {…}
        }
```

Figure 15
Data contract

Figure 15 shows how it is possible to customize serialization. Items can be either a *tDataSetValueResult* element or a *tServiceError*. *XmlElementAttribute* defines what the name for the element will be in either case. If several elements are to be serialized order can define what the order of the elements is going to be.

For the element UUID only *XmlAttributeAttribute* is used, in this case the SOAP elemet will use the variable name.

Depending on the serializer used, the default behavior varices, both in relation to what is serialized by default (public/private) and what the order of the elements are (alphabetical/order of occurrence). For this reason it might not be possible to change serializer unless every important aspect to the communication has been explicitly defined.

[15]- Studies have shown that this serializer also lags complete control over the messages, and DataContractSerializer would not be a poor choice. The way control is done differs for the two.

3.3.3.4 Message contract

Through the use of message contracts it is possible to control how and what is put into the SOAP message, and where. This is an improvement for instance in the cases where some of the data elements are to be serialized as attributes rather than as independent data elements.

Message contracts can be useful in some cases where compatibility with other systems requires that the SOAP message constructed in a specific way.

With all this information WCF is able to create Metadata information for the consumer of the service to use. This can be done either in the regular way where a WSDL is exposed to the client over the web, or the client gets a copy in another way.

3.3.4 Pipeline in the communication

When building a service, the communication from the service to the client and back again is built of elements in a stack. Depending on what elements are placed onto the stack, you can get different attributes for the communication. Elements can be changed, removed or added as needed. This ensures the most flexible communication system possible. Service and client always have to mirror each other's pipeline. When creating a binding for a service, this is where these choices are defined. From a given binding the service can be configured at runtime.

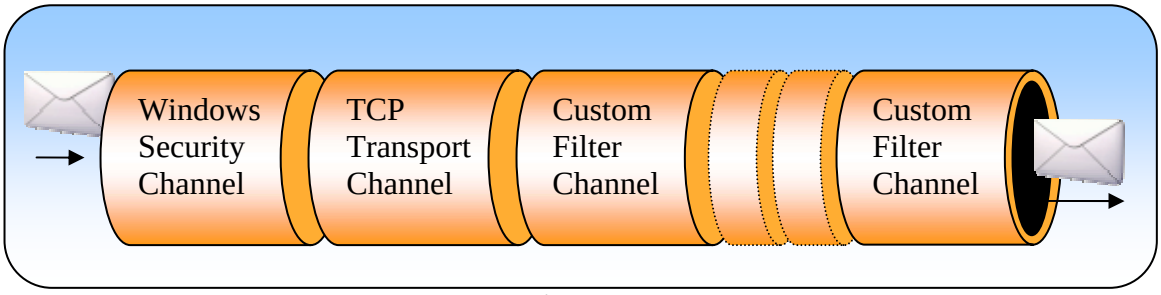


Figure 16
Show the pipeline concerning security, transport encoding and custom filter channel

3.3.5 Custom encoders

Even though there are many standard defined bindings, there might be the chance that an additional one is needed. This is not a problem though. It is always possible to add a custom filter to the pipeline. Of course the filter has to be added to both the sides of the communication to work, but adding custom encoders does restrict the re-usability of the service, because both sides must have the logic for the encoder. Is only possible to specify one binding class in the configuration file, but in code different encoders can extend each other, and encode the message in several ways.

3.3.6 Define the policies using WCF

A WCF service exposes three different ways to accomplish the task

- Code.
- Attributes.
- Policy files.

Code

Usually an instance of a binding is associated with the service. An instance of a class is created and passed as a transport class. This choice gives the programmer complete control over the communication, but it is not a good solution if changes are to be introduced. The code will then need to be recompiled and redeployed.

Attribute

Using attributes, the programmer gets an extremely easy way to add functionality to the code. When a service is working, adding security is simply a matter of adding some attributes to the methods involved. Still the code needs to be recompiled and redeployed, however no actual code manipulation is needed. The main advantage over code is that it is easier to read, and it often requires less lines of code.

Attributes can be involved in many aspects of the code. They are often used to define more specifically how data should be serialized to and from a method in a service. The use of attributes makes the developer's job easier, but because the settings are design time settings, it is not that easy to have more choices.

Policy files

Policy files are XML files defining certain attributes a service should have. The policy file simply contains information about for example the security choices for a service. Both the service and the client can have a policy file. Policy files are a very easy way to change runtime settings for a service. Using policy files separate the behavior of the service from the constraints for accessing the service.

The policy file defines how the environment for a service is used. If the service is moved to another environment or it is decided that the security for accessing the service should be changed, the policy file is changed, but the behavior of the service remains the same. The policy will affect the SOAP messages sent between the service and the client.

In designing the framework, it is possible to build the profiles into it. This would ensure that only allowed profiles according to the standard could be used. On the other hand, incorporating the profile inside the framework would limit the use of other profiles to be introduced later on.

Over time new and better security frameworks are designed, or the security scheme might have to be changed upon the system. It is therefore essential that the secure changes can be implemented in a flexible and easy to use manner. Policy files fulfill this requirement perfectly, and will be used in the prototype.

3.3.7 Hosting

A web service in WCF can be hosted in two different ways. Either it can create a socket and listen for an incoming request from a client, or it can be hosted on a server.

A web service hosted on the Microsoft Internet Information Server (IIS) can benefit from all the built in functionality for handling web applications. For instance IIS has extensive use of security, logging and check on health of the running service. In a large environment with many web applications this will probably be the best solution because the configuration can be managed in an easy way. On the downside an IIS has to be installed on the host machine and of course be maintained.

Self hosting enables the service to run in almost any managed code environment like a consol application or a form application. Self hosting is great for testing or custom applications. To have a service run as a self hosting system will take longer time to develop than a server hosted service. This is because of all the code needed in order to host the service, thread pool, security and service logging. For this thesis it is hosted in an IIS, and self hosting has been used for the tests conducted in case study 3.

3.3.8 Security

Security is an integrated part of WCF and web services. Without security it would be hard to use web services unless you did not care who could read you messages, and who you were communicating to. In a business operation this is usually not the case. At the same time security must be easy to use ensuring that the programmers can concentrate on what they are paid for; to design business related programs.

WCF provides a number of out-of-the-box security features that can be used. Actually, when setting up a standard wsHttpBinding, security is turned on by default.

WsHttpBinding and *NetTcpBinding* both uses message level security by default according to WS-Security and WS-Secure Conversation, ensuring CIA[16] for the message. Credentials is by default set to windows credentials, that identifies the user logged in using Kerberos or NTLM, however, other methods can be used as well. A service can

[16] **C**onfidentiality **I**ntegrity **A**uthentication.

have as may connections as needed; each schema simply needs an endpoint exposing it to the world. [KB2006]

The *basicHttpBinding* do not support the described security features, but that does not mean that the binding can not be secured. In order to have CIA for this binding, it must be deployed over the HTTPS protocol. If the service is hosted under IIS all that is needed is to turn on SSL support for the virtual directory holding the service, and making sure that transport level security is used. Using SSL is a transport based security, and as such it is restricted to an end-to-end scenario instead of a point-to-point security. This makes it hard to reuse the security channel for other transport protocols like TCP, SMTP or MSMQ. At the same time HTTPS do not supply a way to ensure non-repudiation, for a message going through several channels. (Data tampering)

By default an anonymous client is used. This will not tell the server who the client is but it will ensure the transportation is secure through the use of the web servers' certificate. This is the most used pattern for secure web communication today.

WCF provides many security features, and configurations of these securities. These security properties can be changed through the configuration file.

3.3.8.1 Message and transport level security.

Securing the messages for a web service, there exist two major methods. Transport level and Message level .Transport level security uses the underling network protocols for the security. One of the transport level protocols is the well known https protocol. The web service constructs a SOAP message and passes it to the network. Before the message is sent to the network, it is encrypted by the known encryption key. SSL uses a mixture of public and private key encryption, which makes it fast and still secure. Figure 17 shows transport level security.

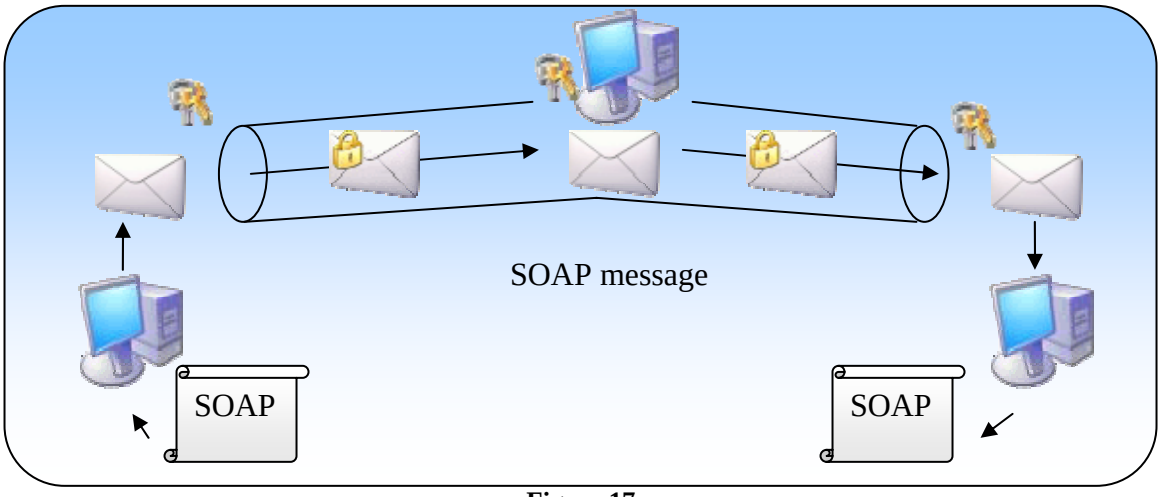


Figure 17
Transport security

The entire SOAP message is encrypted in this way. At the receiving end the message are decrypted and delivered as the original SOAP message. The transport pipe the message goes through protects the message whilst inside the pipe. In a scenario where the message is to be re-routed it has to be opened and re-send on the intermediate base. This of course leaves the message exposed at that intermediate point, making it readable.

In message level security only parts of the message would be encrypted. This would only be the payload. The information about where the message is going to be sent to is completely readable to anyone, but the data is kept secret. In that case the message could be passed freely around the network without any dangers. Only when the message arrives at its destination could it be read. This opens for the possibility of sending the message through several channels. The final receiver of the message is the only one who has the right key to open and understand the message and thus has exclusive access. Message level security therefore reduces the exposure time for a message. Transport level is restricted to certain protocols, while message level can be distributed with any available protocol. Message level security is sometimes referred to as end-to-end security.

Message level security is a newer way to handle the problem, and therefore it support more features for web services security. It is better integrated with the standards for web services, but it comes at a price. The performance for transport level security still surpasses message level security. If performance is an issue, transport level security is then still to prefer. Whenever possible, message level makes for a much better approach concerning the web service standard.

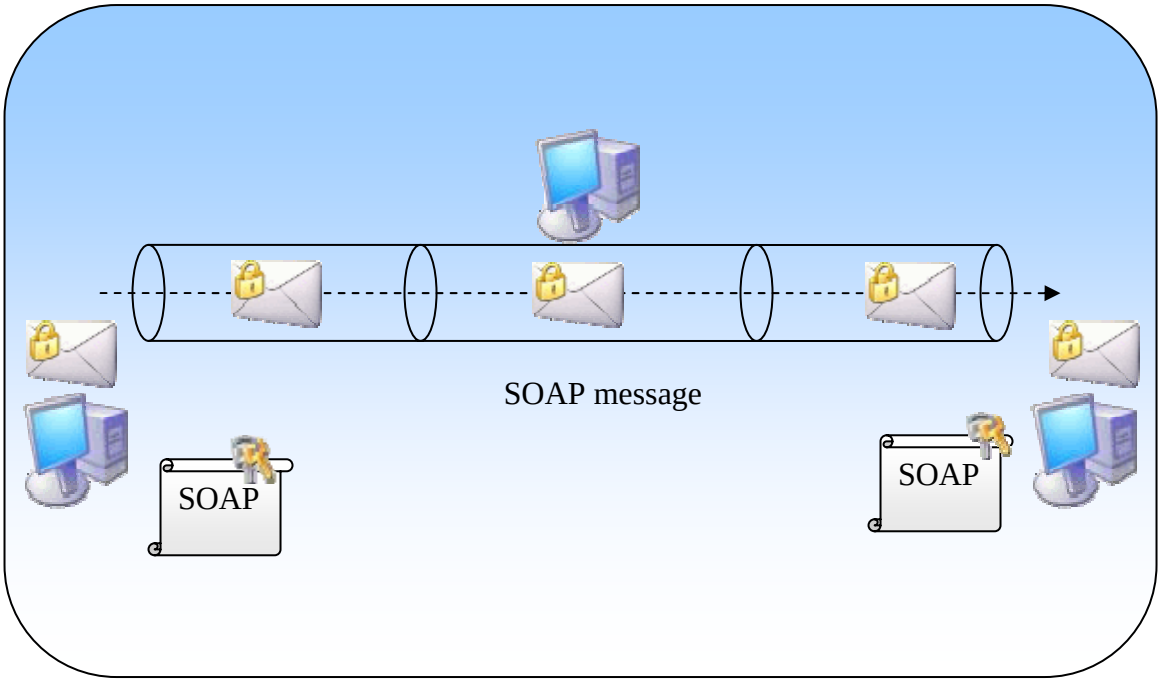


Figure 18
Message level security

3.4 Summary

Different technologies and techniques have been explained throughout this chapter. Especially the new developments in web services and SOA have been introduced and the main advantages of making use of these technologies have been clarified. SOA is to be used in the prototype because of the great cross platform and language inter-communicable properties.

The final product will make use of both WCF and standard web services through a modularized interface. However the main focus will be on WCF since it is a new and enhanced architecture for web services. Because of the way that WCF support WS-I standards it maintains the possibility of incorporating new standards and features in a seamless and easy way.

In the next chapter, a case study for the prototype will be presented where the requirements and the technologies described in this and the previous chapters will be tested.

Chapter 4

IMPLEMENTATION AND CASE STUDIES

The purpose of this chapter is to try out some of the different scenarios that the system is supposed to work under. The system must be able to work under very different configurations, so some of the case studies are of a more general type. The overall design of the system has been built in order to have the same basic design for all the challenges, and then make minor changes in order to add or remove functionalities to the system.

Before studying each case, the setup, overall architecture and the internal workings of the server system will be described in the following section.

4.1 System Description

For developing a good system it is important that all the main aspects of how the system is put together are addressed. For most of the work done in the design, there is not a single perfect solution. No matter what has been chosen as a solution for a problem, several others will have been rejected, maybe not because they were bad choices, but because the overall benefits that they brought to the system was evaluated and found less valuable than other solutions. In this part, different solution for solving the problems, and descriptions for how the system parts have been built to achieve flexibility and stability is presented.

The core unit of the system is a server process hidden behind the service interface where device data and client requests are handled. It has two important interfaces through which it accomplishes its tasks. These are the interface to the communication service which acts as a bridge between client processes and the server, and a second interface which can be configured to pool live data from wind power plant devices. The data retrieved from the configured device(s) is kept in an internal, in-memory data model for fast retrieval and processing when client requests are being handled.

IEC 61400-25 defines an event driven service model for monitoring wind power plant devices. Especially in the reporting process, various conditions have to be monitored continuously as new data arrives, so that proper actions can be taken in order to satisfy each report configuration which will serve the client process with information.

4.1.1 Service Interface - Communication Module

The Service is supposed to work with different client protocols. One of these is Web services, but the system should also be able to work with other protocols like MMS. The service can be thought of as a server communicating the data to clients through several types of communication interfaces.

One solution would be to create a complete, tightly coupled system for each of these protocols. The communication protocol would then be implemented directly into the main server unit, and would make the communication fast because direct access would be granted to the communication layer. On the other hand, this would result in a server bound to a fixed communication protocol. Different clients would then have to communicate with separate servers. Each server would then need to have access to the same data, or would need to have their own copy of such data. This would take up more resources, and changes to the overall model would have to be implemented to each of the servers separately resulting in redundancy.

An alternative method is to have a common server to serve the different communication modules through at well known protocol. A facade presenting a common interface can be created. Each protocol then has to implement a module that takes care of getting information from the client and returning the server reply to the client. This is illustrated in the figure below.

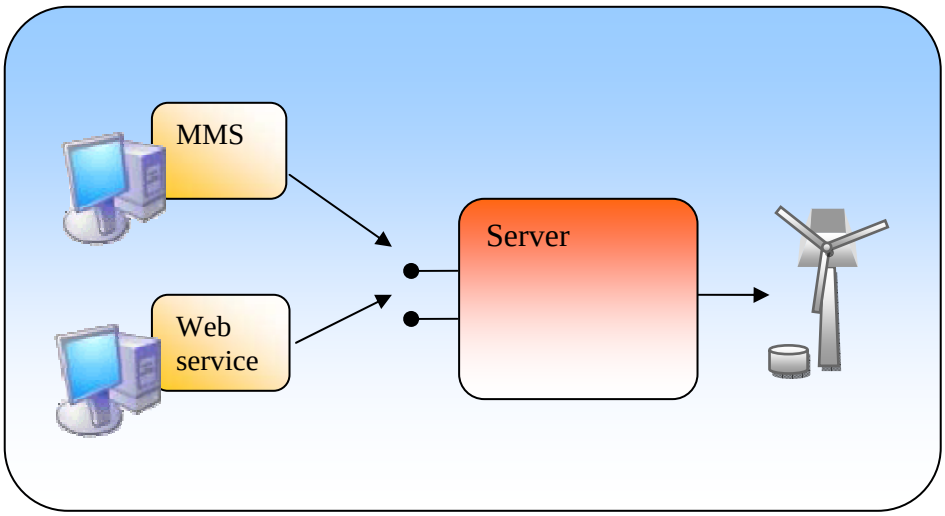


Figure 19
Communication module

In the developed prototype, access to the server is available through .NET remoting. The web service listening for client requests will pass the request to the server and retrieve the response and pass it back to the requesting client process. This approach enables a more modular architecture for the system. The type and structure of the data communicated between server and the web service through remoting is compatible with the services described for each data class defined in IEC61400-25-3. The web service exposing the services of the main server to client processes can be replaced by any communication service using a specific protocol being capable of implementing .NET remoting. That is, the web service is not tightly coupled with the main server which

allows the system to be extended with modules implementing communication protocols different than web services e.g. MMS.

4.1.2 Data exchange between communication module and the server

One major challenge was to make the exchanged data types compatible; not as a structure but on how the different data types are being represented on the different layers from client to the main server unit.

The communication is defined with a WSDL, and is therefore defined as a web service. The different methods in the server and the service are identical, as are the service reply and the response objects. There is however a difference that makes it impossible to use the same data types/classes. In order for the communication to take place using web services, serialization information about the classes must be specified. Unfortunately this requires that all the data must be converted back and forth between reply and response objects.

This is done on the respective communication/service module before passing the data to the main server unit and when a reply is returned from the server. Briefly said, the data being exchanged between the communication module and the main server unit is typecasted so that both the service and the main server can handle data in the types known locally.

4.1.3 Device Interface

The communication interface to wind power plant devices from the main server is out of scope for the IEC61400-25 series. No specification is given on how to handle the communication with the devices that are supposed to be monitored and controlled. However, in order to be able to test the server prototype a simple simulator is implemented and connected to the system. The simulator acts as a device continuously generating data for a specified set of data objects, for a pre-specified logical device data model defined on the server. The data is extracted from log files which contains real-time stamped wind power plant device data. The data entries contained in the log files are mapped to data types defined in IEC61400-25 where suitable.

The simulator process is only capable of producing data; it does not model and simulate a complete wind power plant device with all its components and functions. Therefore the simulator is not capable of representing any data being set by the client process at the other end of the system. The server only assumes that data is sent and committed by the device simulator. That is, the data sent from the client process will update the internal data model on the server where the server assumes the data has been passed to the wind power plant successfully.

Communication between the device simulator and server is implemented with .NET remoting. The simulator is set up with a server role, reading log files and passing the data over to requesting clients through .NET remoting. The server acts as a client with respect to the simulator. At startup initialization the server will connect to the configured simulators through device connectors and periodically request for available data and update its internal data model, and make the data available to client processes.

This part of the implementation is merely a solution to be able to test the functionality of the server. It is not a proposed solution for a real life problem where wind power plant devices have to be connected to a system both for data retrieval and remote controlling.

4.1.4 Server Configuration

Initially at startup, the server must be configured in order to work properly and interact with devices and communication services attached to it. At startup it will configure itself through an XML configuration database. The information model reflecting device data, device connections, pre-configured data sets and access control configurations are all made through XML files dedicated for each specific configuration.

Configuring the information model

At startup the server will first create the data structures reflecting each specific device attached to it in order to be able to house the data and make it available to client systems. The XML configuration file makes it possible for a server administrator to create data structures for logical devices with its logical nodes, data objects and data attributes. Initially the configuration file defining the data structure with their default values is parsed where afterwards the object instances are created on the server according to the structure defined in the configuration file. It is possible to initiate data structures for several logical devices in the same configuration file.

The configuration file resides on the server's directory which implies that the server administrator must know the exact data structure of the physical devices which are to be attached to the server, before composing the configuration file. In the future the configuration could be done dynamically by retrieving configuration information directly from the physical device itself. The figure below shows a short snippet of the XML file used to configure the data structure on the server.

```
<?xml version="1.0" encoding="utf-8" ?>
<Server>
  <LD LDName="WPP1" LDRef="WPP1">
    <LN LNName="LPHD">

      <!--Physical device name plate-->
      <DATA DataName="PhyNam" AttrType="WDPL" Presence="M">
        <DataAttribute DATName="vendor" DAType="visibleString255"
                       FC="DC" TrgOp="" Presence="M">
          <value>Vendor information goes here.</value>
        </DataAttribute>
        <DataAttribute DATName="tmOffset" DAType="int16u"
                       FC="CF" TrgOp="" Presence="M">
          <value>0</value>
        </DataAttribute>
        <DataAttribute DATName="tmUseDT" DAType="Boolean"
                       FC="CF" TrgOp="" Presence="M">
          <value>false</value>
        </DataAttribute>
        <DataAttribute DATName="tmDT" DAType="Boolean"
                       FC="CF" TrgOp="" Presence="M">
          <value>false</value>
        </DataAttribute>
      </DATA>
      ...
```

Figure 20
Server configuration – data model

Configuring device connectors

Currently only device simulators can be attached to the server through device connectors which will be described later in this chapter. The server makes use of .NET remoting technology in order to communicate with the device simulators. The XML configuration file defines all the device simulators which will be attached to the server. At startup the server will parse the configuration file and establish connections to each of the configured device simulators. The figure below shows a short snippet of the XML file used to setup device connectors.

```
<?xml version="1.0" encoding="utf-8" ?>
<DeviceConnectors>
  <Connector>
    <LDName>WPP0</LDName>
    <TCPRemotingConnString>tcp://host1/WPP1</TCPRemotingConnString>
    <PoolInterval>10000</PoolInterval>
  </Connector>

  <Connector>
    <LDName>WPP1</LDName>
    <TCPRemotingConnString>tcp://host2/WPP2</TCPRemotingConnString>
    <PoolInterval>10000</PoolInterval>
  </Connector>
  ...
```

Figure 21
Server configuration - device connectors

4.2 Case study 1

The case study will cover:

- Deployment of 2 different WPP data suppliers
- Securing data
- Securing access
- Cross language access.

In this case study a small community has put their savings into a couple of WPPs. They have had one WPP for some years now, and have now decided to buy a new one of a different kind because the old model is not produced anymore. When they got their first WPP they also got a simple client system for monitoring purposes. The owners are not interested in getting a new fancy system, they just want to continue to use the same client system they have been using for a long time. Of course they would like to view the features that the new WPP supplies, but the most important thing to them is that they can keep on using the system they know.

They will have two different kinds of WPPs, but because both models support the IEC 61400 standard, they know that the new kind will still work with their old client system and vice versa.

The owners view the data over the public internet, and are using several kinds of security techniques. The standard client is running on an Internet Information Server (IIS) which makes it possible to inherit standard security features.

Aeolus is one of the owners, and he is interested in computer science. He would like to create his own client. The standard server came with a DLL encapsulating the communications interface with the service. But he prefers to use java instead of .NET. He decides to create his own Java client for communication with the service.

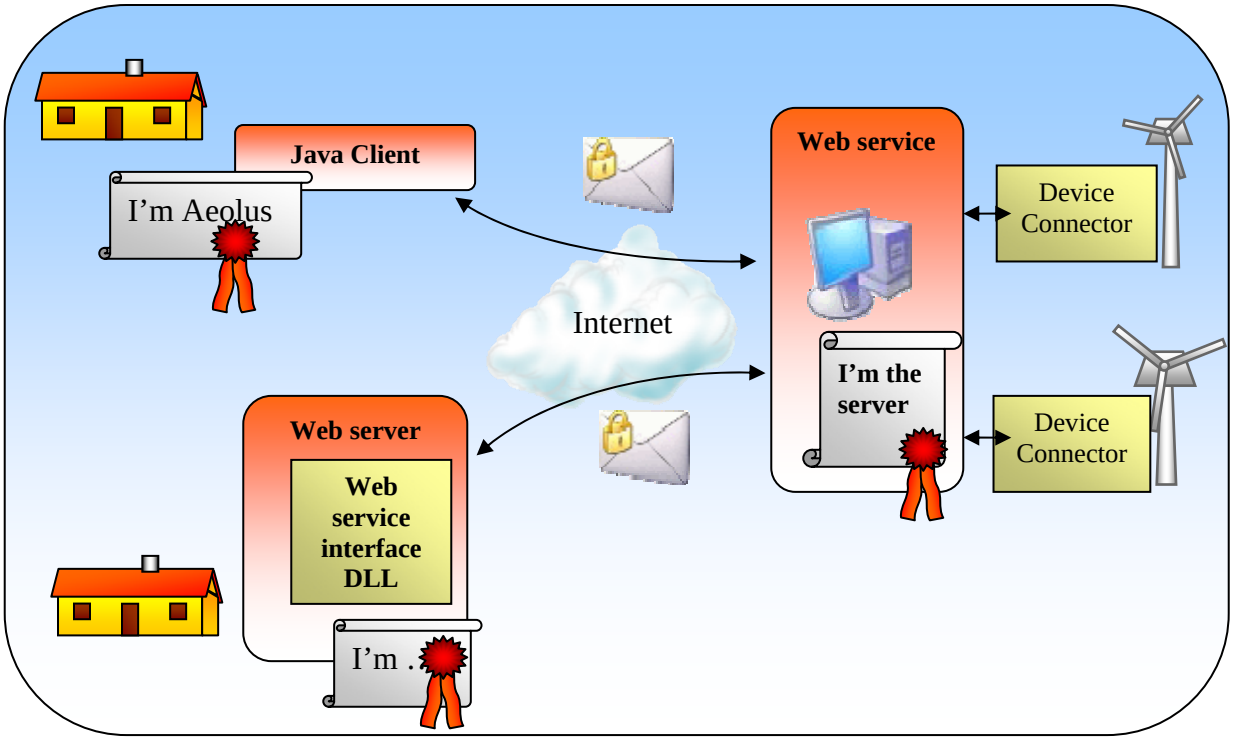


Figure 22
Communication between service and client using different platforms

Multiple device connectors

The server is running and has been configured for the two WPPs. The server needs to retrieve data. This is done through device connectors. The device connector has to collect data from the WPP and convert it into a format the server can understand. The IEC61400-25 series does not specify a protocol for device communication, Therefore a simple protocol must be designed for the WPP simulator.

One possible solution would be to send the data XML as it is defined in the example in IEC 61400-25-4 on page 24. The logic for this would be clear, but this will send a lot of redundant data. To store the data, either it would have to be converted into a more compact format, or it would take up a large amount of space.

Another solution is to send the data as plain-text reference-value pairs where a timestamp is attached. This will reduce the amount of data being transferred, and still be able to transfer data values with the correct semantic.

```
Timestamp Wpp1/WGEN.Spd.mag=value
Timestamp Wpp1/WGEN.Spd.mag=value
.
.
.
Timestamp Wpp1/WMET.MetAlt1.HorWdSpd.mag=value
```

Figure 23
Reference path and values

Data is still readable, and does give a much quicker overview of what it actually contains in contrast to a huge XML file. This is by far the best approach, and this has been chosen.

In order to design how the information is exchanged on the message level it must be specified how the architecture of the system is defined and set up.

There exist at least two different solutions for how the architecture can be defined. Figure 25 shows the first case where the server and device connector are located on different machines. The WPP has a system running under .NET, where the data is exchanged over a network. One of these is through distributed objects using .NET remoting object. This is a transparent object interface where distributed objects hosted from different application domains can be accessed (similar to the Java RMI technology). .NET remoting is similar to web services; however it is using non-standard binary communication, and is therefore restricted to the .NET platform. In this case remoting has been used in implementing the device connectors. Different communication protocols could be used by replacing the server's device connector module.

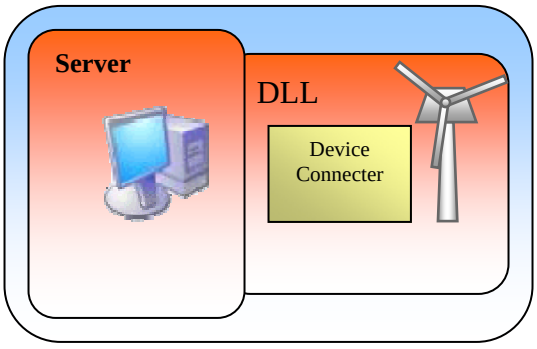


Figure 24
Device connecter and server on the same computer

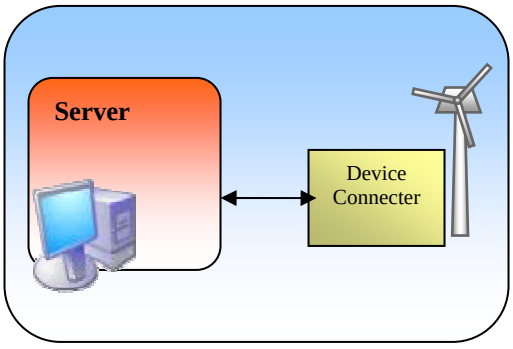


Figure 25
Device connecter and Server separated

In another case the server is placed at the WPP as shown in Figure 24. In this case the server has direct access to the data. The device connector could be compiled into a DLL which is directly capable to communicate with the device.

The Prototype uses the first solution, because the server is considered as a central unit able to manage several WPPs.

Building the device connectors

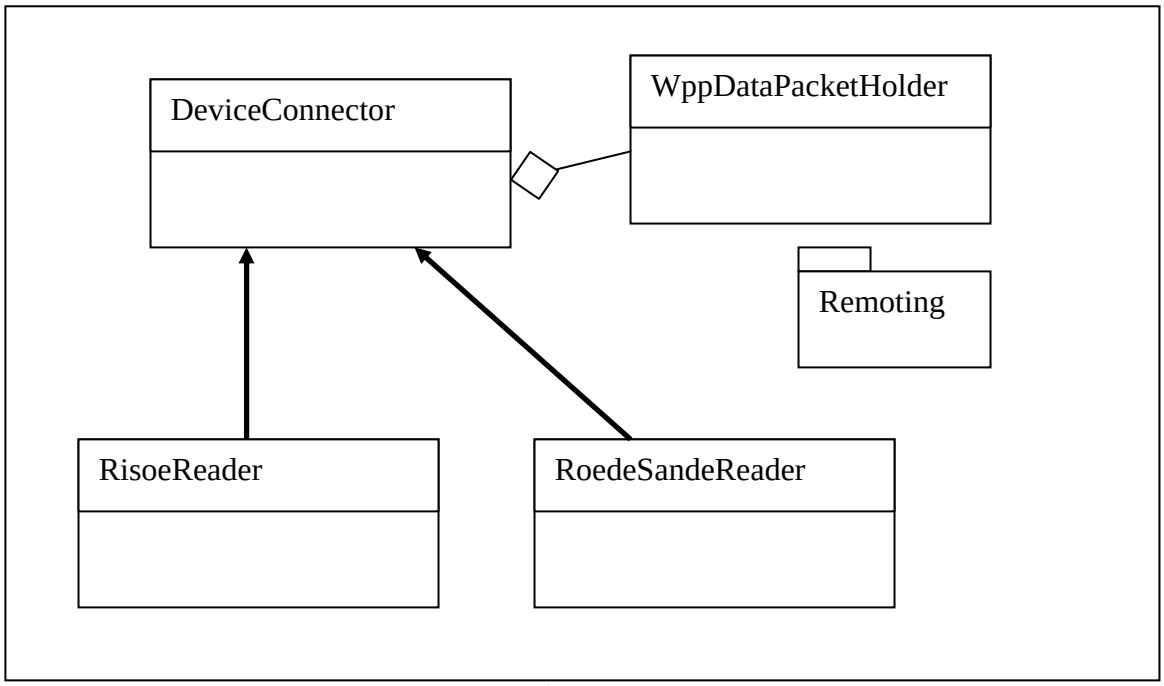


Figure 26
Class diagram of device connector classes

The device connector follows a common pattern no matter how data is retrieved from the WPP. A communication line between the server and the WPP must be created, in this case where remoting is used, this is comprised of setting up a remoting stub that can be posted to it clients. The device connector must also take care of a data storage buffer that can be accessed by both the connector itself and the server connected to the device.

DeviceConnector is an abstract base class setting up the remoting and the internal storage. *RisoeReader* and *RoedeSandeReader* each inherits from this class. When some data has been produced it is placed in the storage for later consumption of the server.

It is not clear if each WPP should have its own data connector. The data connector can gather data from many WPPs but for the time being, the server takes for granted that each connector represents one WPP.

WppDataPacketHolder holds data in a list. The server asks for data within a specified time interval. All the data in the buffer is delivered on a request, where it is ensured that data is not lost in case of a connection failure.

Both the data connector and the server are trying to access the storage at the same time. This is a classic producer consumer problem, and the access to the data storage is blocked by the process that is either reading or writing data to the data buffer, to ensure that the integrity of the data is not compromised.

The Diagram shows the functionality of the data pooling system in a generalized way. Please note that the diagram shows two processes sharing a synchronized data storage device.

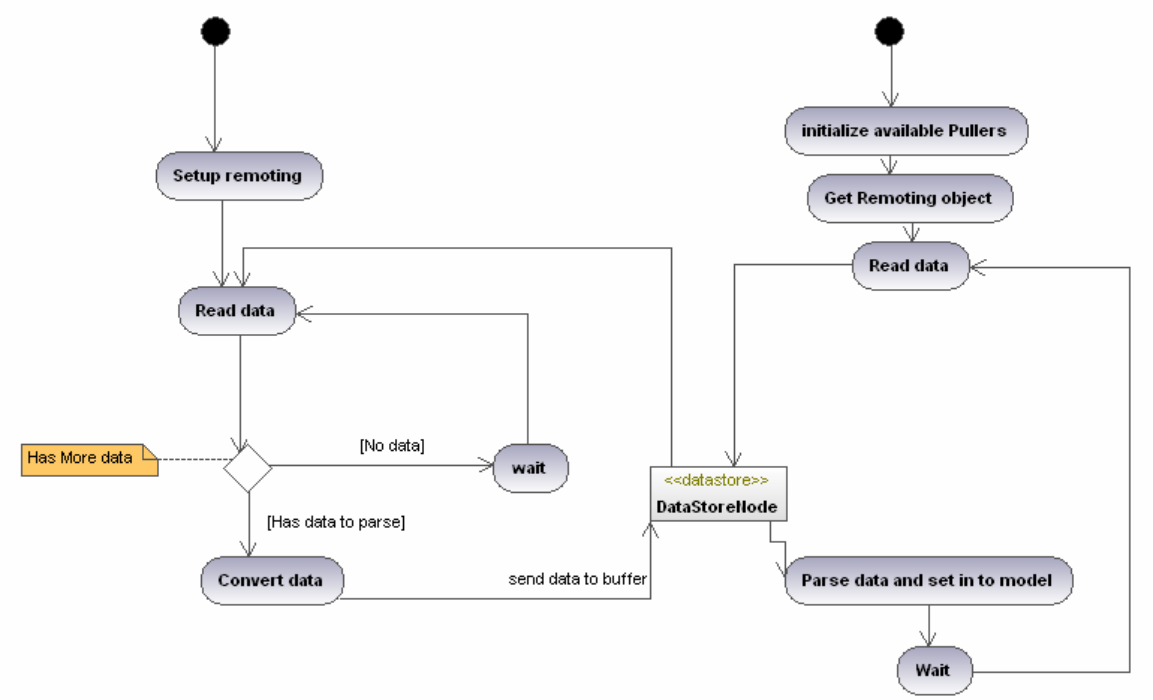


Figure 27
Flow for device connector

4.2.1 Java client, C# Client and DLL as a client

The Java client is a simple terminal application where the user can type in the commands (mapped to service calls) and view the results sent back from the main server via the web service. The Java client is implemented using Apache AXIS. This is a Java API to develop and deploy web service applications. The purpose of implementing a Java client is to prove that the service is able to interact with clients developed for other platforms than .NET. The client is not intended as a fully functional client, but merely as a proof that the communication is possible. Therefore the Java client does not contain all the available methods, and is not tested as extensively as the windows client.

Independent of how the clients are going to communicate with the web service, they must have a copy of the contracts. Either a copy is retrieved from the server, copied to the host, or retrieved at runtime. Information about the service location and contract can either be known beforehand, or a UDDI can relay this information to the client. To interact with the service a proxy class must be created. The most common way is to create a static proxy; however a proxy can also be generated on the fly. In this case the proxy is statically generated by retrieving meta information form the web service.

In WCF, metadata exposure is not enabled by default. An endpoint using the *mexHttpBinding* binding exposing the contract *IMetadataExchange* must be created. A call from the client can retrieve the WSDL for the service, from where the proxy can be generated. The internal construction and how it is done depends upon the tool used, but the overall construction is based on the same pattern. AXIS generates a proxy class for the contracts interface, each request, and reply type in reference to the methods. Data types used in the methods are placed in classes. These classes specify how the data are to be serialized and de-serialized as SOAP objects.

In order to establish a connection a class *IEC641400_25ServiceLocation* is created, this class contains the logic for creating the binding. The class has different methods for each endpoint the web service contains. If the contract changes, the methods and the data types used must be changed as well.

Calls to the service are done by creating an instance of the proxy object, and calling the service method through it.

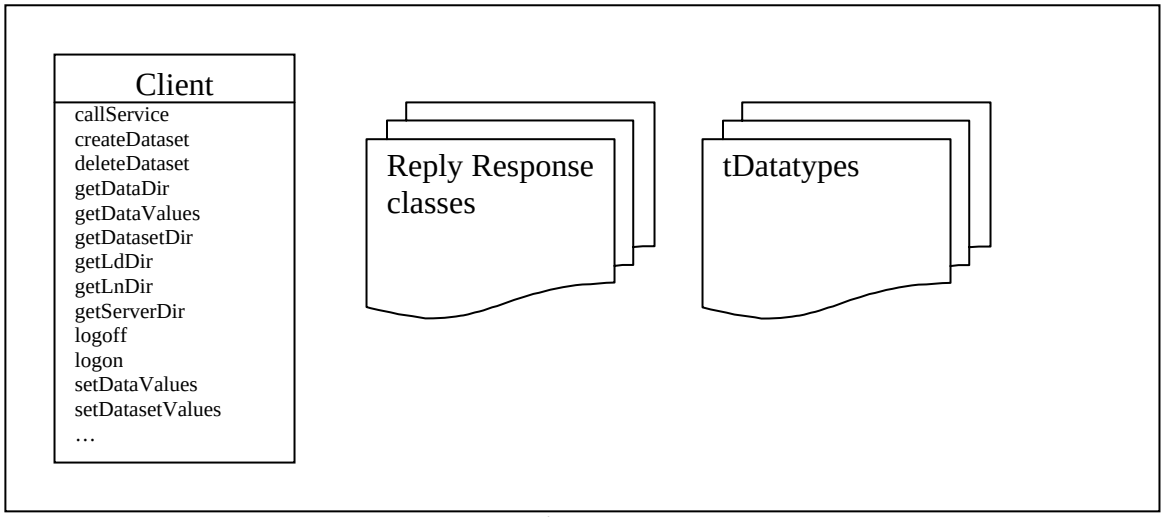


Figure 28
Class in the proxy

The way the Java client was created is very similar to the way a WCF client can be created in .NET. A proxy is created and is used for the communication.

Unlike the Java client, WCF shows its strength here. Setting up the bindings and other attributes for the service and the client can be done in their respective configuration files. The proxy is compiled into a DLL, which can be distributed to different clients. To use the service, all that is needed is to include the DLL into the application runtime and create an instance of the proxy class. For the prototype the configuration file is used for setting up the binding for the client. Even though the logic of the methods is encapsulated in the DLL, using the configuration file still enables specifying different settings for the client. The information and settings could be embedded in the DLL by using a facade pattern, but having it in a separate configuration file ensures flexibility for changing the

settings whenever needed. An important point is that changes to the client's configuration file must be reflected on the server configuration; otherwise the connection will be rejected.

The implemented WCF client is a simple windows console application making use of all the available service methods exposed through the proxy instance.

4.2.2 Presentation of data

A terminal client might be a fast way to present data to the user; however it is not suited for end users. To demonstrate the capabilities of the prototype a web client making use of visual components would be more suitable.

An ASP.NET project has been running in parallel with this thesis, where a web client for the prototype has been developed. The communication between the web server and the web client is outside the scope of this project, and no measurement has been taken to secure the data communication.

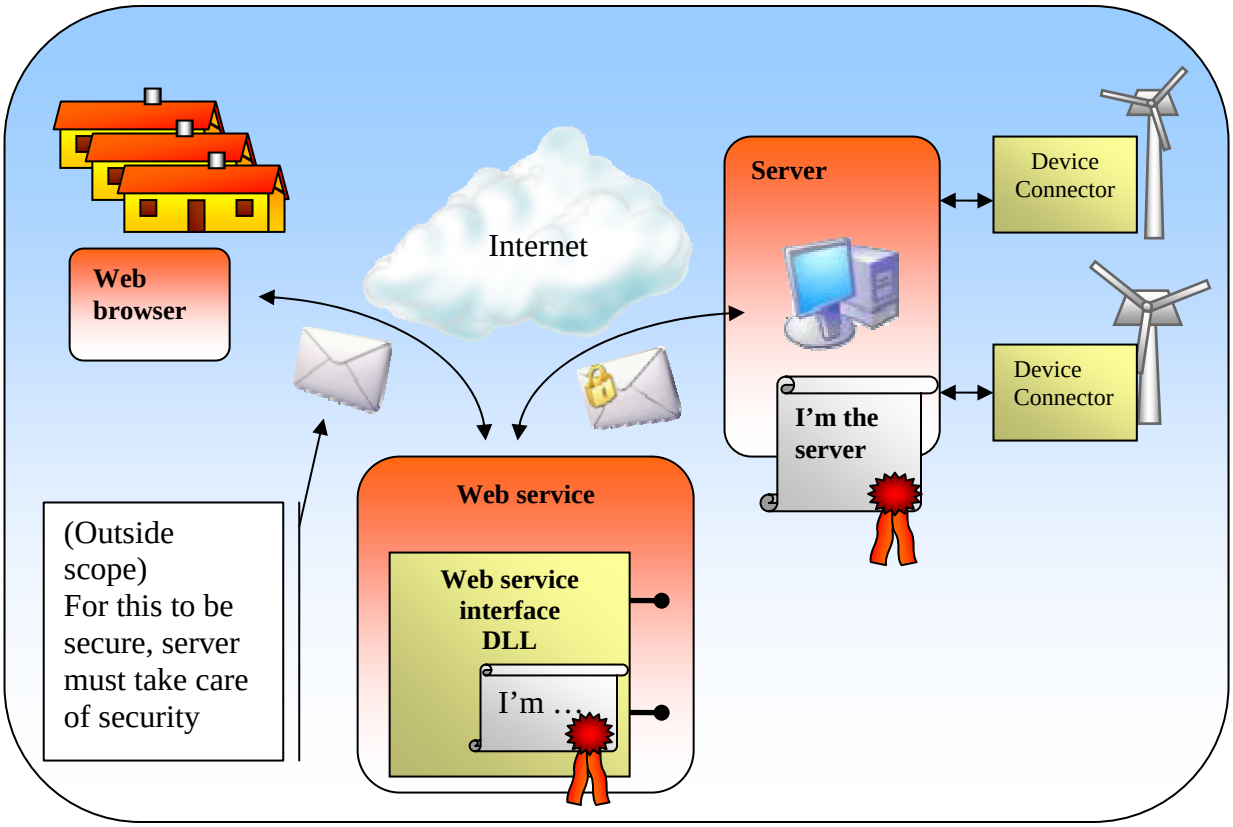


Figure 29
Client communication through a web server

4.2.3 General decisions in securing the system

Setting up a secure system is not a simple task. Many different approaches can be used, each of them having a different impact on the systems performance and different implementation issues. The system is built up of a number of small modules, which communicates across boundaries in several places. A boundary is defined as communication between processes not sharing the same application domain/runtime. Some of these boundary crossings include communication over a network protocol. Both inter-machine and inter-process communication can use network protocols for their communication. For the system to be secure, every one of these communication lines must be secured. The system as a whole is only as secure as the weakest link. A single insecure line of communication could render all other security aspects useless. The different combinations of the modules located on different machines would have different implications on the overall security of the system. .NET remoting might not be a security problem if it is used between processes running on the same machine. Otherwise if it runs over an open network, the communication should be secured properly. This can be done by filtering traffic and encrypting the communication as discussed in 3.3.8.

In this thesis the focus is on web services, and security has not been dealt with from end to end, but more on a level showing some of the possible approaches which can be taken. Depending on the deployment of the system different approaches must be taken to ensure secure end-to-end communication. For this reason analyses must be conducted every time the system is deployed in a new module configuration. And measurements must be taken for each boundary crossed. Figure 30 shows all the possible points at which security must be addressed. If the communication is done on the same local computer the list can be reduced.

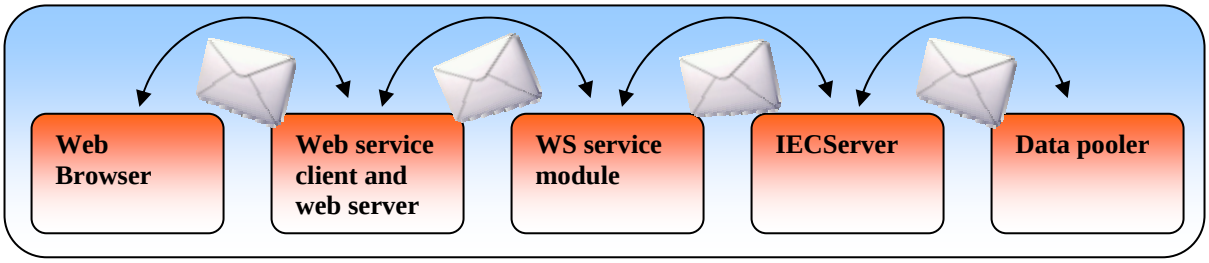


Figure 30
Worst case scenario of boundary communication

For the developed prototype, security includes the basic CIA (Confidentiality – Integrity - Authentication) for the communication.

As described earlier, WCF supports several standard security schemes. Most of the cases are centered on exchanging key(s) securely.

One of the widely used scenarios involves a certificate(s) (Cert) for the key exchange. In this test case mutual certificate exchange is done for both client and service. The client must present some credentials identifying itself to the system and the server must identify itself to the client. When this is done the system must make sure that the

communication upholds **confidentiality** (encryption), **integrity** (not tampering with information by malicious persons), and **authentication** (ensures that claim of identity is true) for the information transport. This can be done in several ways, but the combined package comes together in a Certificate, which can be exchanged in a standard way using the International Telecommunication Union (ITU) standard X.509. A certificate contains information about an owner and a keyset used for asymmetric encryption. WCF supports mutual X509 certification by supporting both WS-security 1.0, and WS-Security 1.1.

The use of mutual certificate exchange ensures that both client and server know who they are communicating with. The data exchange will take place only if both parties accept the presented credentials. The certificates used for this case study are signed by a mutually trusted certificate authority (CA). For the use of the prototype server this proves to be a valuable approach because it will restrict access to the service to authorized parties only, and at the same time it proves to be able to perform fast communication. However it does require more work in order to set up the system. The lighter versions of this scenario are single certificate exchanges (as in SSL). This provides the same features concerning traffic security. However, both server and client must be identified in an alternative way.

The appendix holds a short explanation of how to set it up and use certificates in WCF. The glossary contains a description about the content of a certificate.

Case Summary

This case has shown that web service clients in both Java and WCF can be connected to the web service, and retrieve the information gathered for different WPPs. It is also shown that a web client can gain access to the service though the use of a web server.

The device connectors could act on their own or represent several WPPs. The gathered data could be collected and managed through one central server.

Security in the system must be addressed on many levels, and must be analyzed for each case.

4.3 Case study 2

Deployment of a wind farm

In this case study the focus is on a virtual major corporation owning a big wind farm. Combined of offshore and land based power plants. They operate many WPPs and are monitoring the system from one or more locations. The offshore WPPs are connected using a modem. For this reason it is important to get the most out of the bandwidth by compressing the data transmitted. In this scenario, the communication line between the WPPs and the server can be lost. In case of connection loss it must be possible to recover the data produced during the connection loss.

Some of the WPPs have their own device connectors and others have a joint connector. Within the corporation many people must be able to view the data at the same time. The corporation has a dedicated secure private line for their communication. No security is needed concerning the securing of the traffic. However, data must be protected internally. Not everyone in the corporation is allowed to access data, and even if they are allowed, not everyone (or application) is allowed to run all service methods on the WPP devices.

Key points:

- Many WPPs on a service.
- Multiple clients.
- Avoiding data loss during offline period.
- Compression for smaller bandwidth restrictions.
- Role based access control.

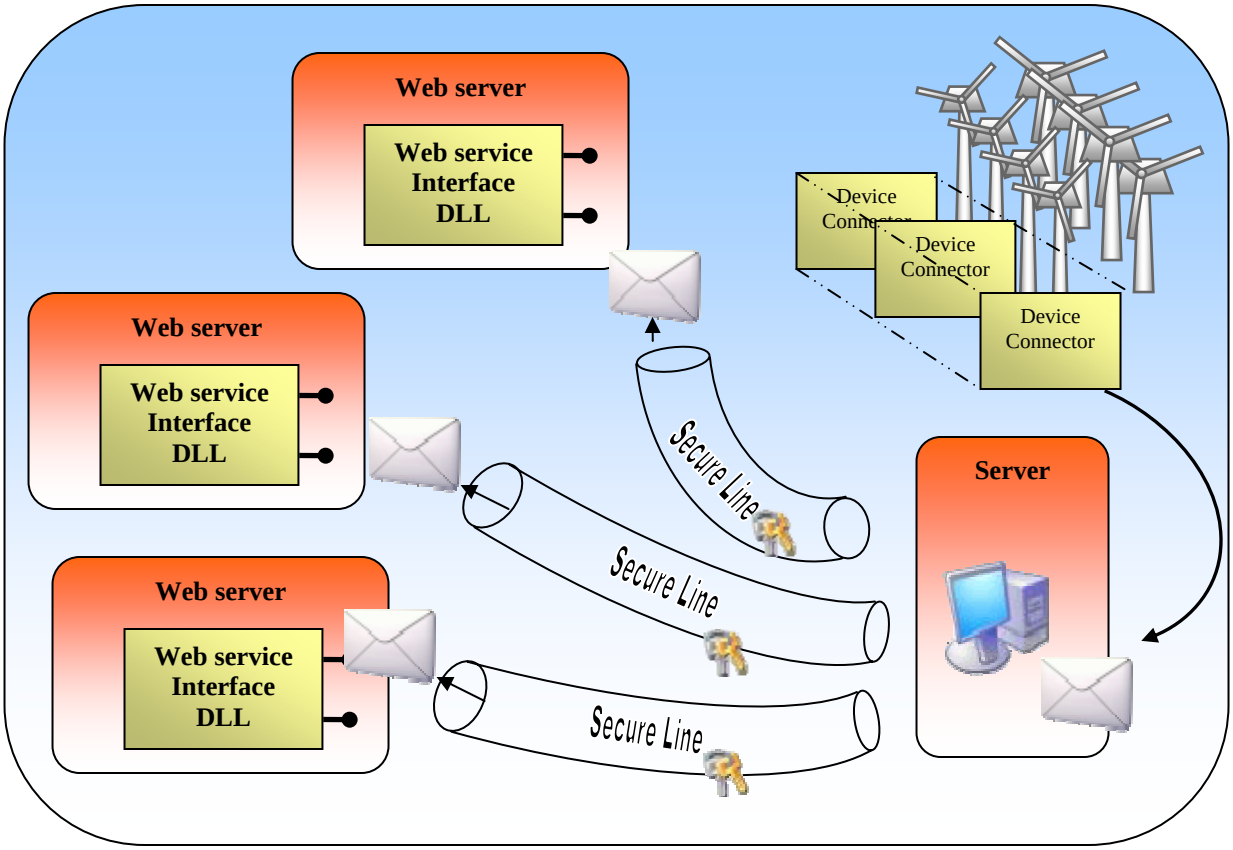


Figure 31
Multi WPP and clients

4.3.1 Secure connection

Because the communication is done on a secure line, no additional measurements must be taken in order to assure traffic safety. Having a secure line actually reduces the possibility of one of the boundaries leaking information, because security can be wrapped around the entire system. Virtual private networks among others provide a closed secure line.

4.3.2 Offline scenario

Data must be delivered to the server one way or the other. In the case of loss of connection, data must be stored until the client can retrieve it. An extra or redundant server connection could provide an alternative way to upload data, but since WPPs are supposed to be autonomous and recover from critical situations independently, it must be concluded that information is not time critical at a level where it will justify the extra cost of maintaining several communication lines. A much cheaper method is to use the hardware already placed at the WPP. In such a case each device connector keeps a buffer of data, which can be retrieved on demand. However, if the data is not retrieved by the client, the list will keep on growing in memory and eventually consume all of the memory resources.

To overcome this problem data, must be swapped to disk on regular intervals. This will not only reduce the memory usage, but it will also assure that data is not lost if the machine hosting the data connector crashes.

The device connector works with time stamped packages. If the buffer holds more than a given number of entries, the data will be saved to disk, and the stack will be emptied. On the next attempt by the server to retrieve data, the retrieved object will contain a flag telling the server that buffered data exists. The server can retrieve this data on demand.

This solution is extended in order to avoid problems in some cases. First of all every new addition to the file requires that the whole file is read and written, due to the way serialization works in C#. With small files this is not a big challenge, but if the file becomes really big, the task will be time consuming. A new file must be created if one file gets too big. Another potential problem is if the data set is too big for the service to serialize it. In the binding it is possible to specify that data should be either streamed or buffered, which would solve the problem. However, if the device connector has gathered data over a longer period of time, the server must be able to retrieve the data in smaller chunks. For the device connector it is possible to define the desired file and buffer size.

If the connection is poor but do exist, reliable sessions is a great addition to the quality of the communication. “Reliable sessions” or “reliable messaging” assures that a message will be resent if the communication is disrupted. It will keep on trying until the message has been delivered, but at the same time uphold an assurances of exactly-once and in-order delivery of the messages. Reliable messages is part of several of the standard bindings, and can be turned on and off on demand. Like other attributes this can be done either in code or in the configuration, by enabling reliable sessions. Reliable messaging is turned on for the prototype.

Using reliable sessions alone does not guarantee end-to-end reliability; it merely takes care of the web service traffic. Additional measurements must be taken between the server and the WPP to assure traffic reliability.. How this is going to be done depends upon the communication between the two are done and is as such out of scope for this project.

4.3.3 Access control (AC)

To ensure that only the right people have access to data, an access control system is needed. The standard proposes an access control system, but for the time being it is not in its final stage, and will most likely change when the standard is finalized. For the purpose of the prototype, a simple system for controlling the access to the system has been implemented. This is built as a role base access system.

Instead of providing rights to every single user; each user gets assigned to one or more groups instead. Users are members of one or more groups, and can be granted access to a node or method through rights assigned for groups. Whenever a user tries to

gain access to a service or a data instance, the system will check if the user has the required privileges.

The user can also have personal access rights assigned. It is always the personal rights that have the precedence over rules given to a role. User rights can be both positive and negative. This is done in order to remove rights for an individual, who had elsewhere been granted access to a resource by a role he is in.

The user identifies himself to the system where a principal object is created. This object is initialized with access rights for a user. The user is identified through the use of a username and a password, but any other claim authorization technique could be used. When the user tries to access a node or service, the name of the node or service is checked against the user's principal object. A positive answer will allow access. Nodes are checked in the same manner.

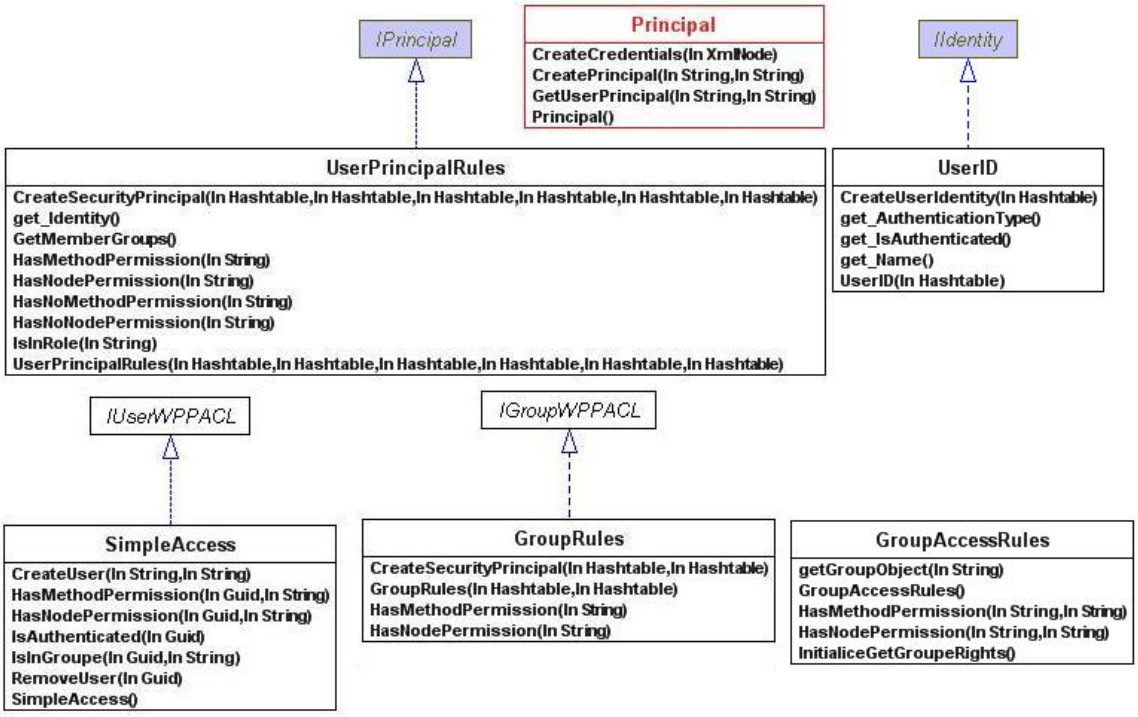


Figure 32
Class diagram for AC

SimpleAccess	Controlling object, containing a list of all the active users and dispatches security questions
GroupeRules	Class holding permissions assigned to a group

GroupeAccessRules	Class holding access rules for each group
UserID	Class holding id for user
Principal	Class creating a principal object for a user
UserPrincipalRules	

For this test, two XML files contain the users, roles and access information. For the final system, this information could be moved to a relational database. But for the prototype it is faster and more illustrative to edit an XML file. Description of the files can be found in the appendix.

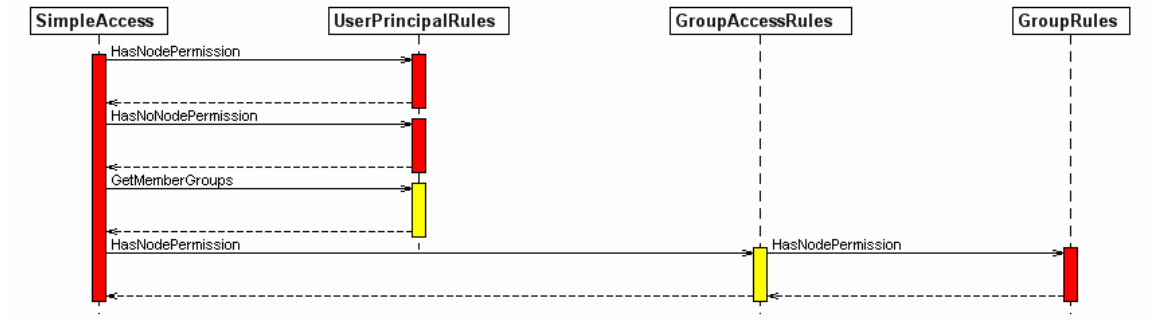


Figure 33
Node access permissions

The sequence diagram in Figure 33 shows the order of the access checks. A user must have personal access rights to the node, and he must not be disallowed access to the node. If neither of the checks resolves to a positive result, each group the user belongs to is checked for access rights.

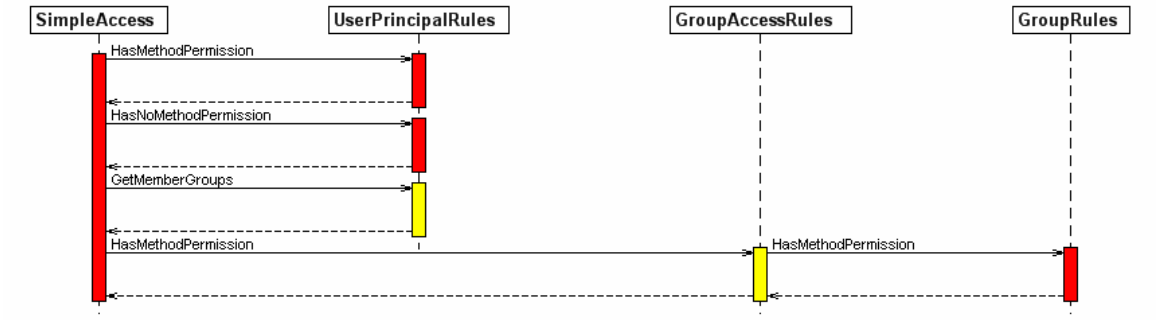


Figure 34
Method Permission

Access for a service is done in the same way as with node access. Service permissions are shown in Figure 34

Method Permission. The AC can be expanded when more precise rules have been defined.

4.3.4 Ensuring module exchangeability

The modules in the system should be interchangeable. For instance the AC module uses a specialized way to figure out if a user has access to a node or not. The described AC uses a role based schema using XML files. But what is needed in order to replace the module with a schema getting its information from a database. The module is accessed from many different methods in the server, and as such it is incorporated deeply in the system. The solution lays in linking the object during startup rather than under compilation. The principal is described by Martin Fowler as a plug-in pattern. [MFPEAA]

The AC module is placed in a separate compiled module, ensuring that other types of security can be used if preferred. As long as the access module uses the access control interface supplied, different security schemes can be used by simply replacing a single file.

When the server is started, it looks at the assembly of DLLs in the runtime directory via reflection. If one of the modules implements the interface an instance can be created and imported into the current assembly, and have runtime binding.

This pattern is one of the corner stones in order to have interchangeable dynamic modules. Negative effect is that DLL access is a little slower than if the linking was done at compile time. But this is a minor expense compared to the big flexibility it provides for the system.

The downside to this approach is that all modules must conform to the same module interface, and therefore the interface must be finalized and take into account that different security patterns need different access methods. For the prototype the interface has been designed for a username and password authorization pattern. On the other hand if the module was incorporated directly into the server, the code would also have to be rewritten and recompiled in order to execute the change. For this reason the interface choice does provide a better solution.

The plug-in approach should also be incorporated into the device communicator, removing the need for tying the system on a specific communication technology. Also the storage module is a candidate for this pattern.

Minimizing traffic

SOAP is not a good format when considering the amount of traffic generated. It is in plain-text format and contains lots of meta-data. This makes SOAP a perfect candidate for compression.

For web services using WCF there are three main ways to encode the data for transport. These are text, binary and MTOM. Text is plain SOAP in ASCII encoding; this

of course is the most compatible way to send the data, but also the most expensive in terms of bandwidth consumption. MTOM is a WS standard, and compresses the body of the message. MTOM can also be used for transporting binary data between services. This comes in handy if files should be transported from a server to a client, but since it still have to be transformed to base64 encoded data and still uses SOAP wrapping, alternative methods like FTP transport would probably be a better solution for this. Different bindings or compressions do reduce the traffic drastically, but they do slow down the system because of the extra computation needed.

Binary traffic is the most optimal way to transport the data, but is restricted to windows-to-windows communication. However if the communication is restricted to a Windows-to-Windows environment, this is the best choice. In Case 3 a study of the byte amount passed on the wire has been conducted.

The last approach for reducing traffic is to create a custom encoding schema. For instance, the traffic could be encoded using one of the commonly used compression libraries, like ZIP or RAR. This of course will demand that both the client and the service are able to encode and decode the traffic. Case 3 takes a look on a custom encoding stream using GZIP. This is a custom binding created for test purpose, however, it can be used in the communication.

Simulating the wind farm

As described in the previous case study there was designed two different device connectors. The first one produced data for several WPPs while the other only produced data for a single WPP. In order to test systems capabilities to manage several WPPs, a number of device connectors have been initiated in separate threads. In the server's configuration file, each WPPs device connector was registered under its URI address that the connector is exposing its remote objects on. According to the configuration, the server will periodically ask the WPP (through the device connector) for available data.

In the other end, clients were created in threads to invoke service methods and retrieve data form the system.

Case Summary

This case showed how a device connector module is able to store data for later retrieval from the server. The module provides a way to get data whenever possible, in manageable data packages. This ensures data delivery, and at the same time reduces peek loads on both the device connector module and the server. No standard is defined for how the data is conveyed from the physical device to the server, but the connector (device simulator) did provide some basic cases that all connector should address no matter how they are designed internally. The lack of a standard for a WPP communication makes it hard to reuse the system between different vendors, because the module will be incorporated into the server, unless a common interface is defined.

As described with the AC module it is essential to define an interface if the module is going to be separated from the server. However, with a defined interface at hand, the system would be very flexible and have opportunities for changing the modules.

The case was a proof of concept test. The simulation showed that it was possible to have several clients and multiple WPP simulators gathering data at the same time. Feather studies should be made in order to establish the performance of the system; however this has not been conducted due to the lag of real data.

4.4 Case study 3

This case study looks at some of the different bindings and protocols that can be used in the communication. This is split into two main cases, the first one looks at the number of times a method can be called in a time span of 10 seconds where the other looks at the amount of data sent. This can be used as an indication of what the different choices entail in relation to speed and throughput of the system. For instance, is the number of handled calls a service can be called much smaller if the communication is encrypted, and how many fewer bytes is sent by using binary encoding compared to plain text/SOAP encoding. The test also has the very important task to prove that it is possible to create and use different endpoints for the same service, where the configuration differs.

It is worth remembering that most of the bindings support the same basic features like security, reliable messaging, different encoding schemes, and as such the basic features can be kept no matter which binding is chosen.

How to design an implication test

If the tests are to be comparable it is essential to be able to control the system parameters individually. Tests with the same parameters are conducted, where one parameter at the time is changed. This will give a clear indication on how the different configurations will affect the system performance.

Dealing with a system comprised of web services, it can be hard to give a specific answer to how much a given choice influence the entire system. First of all it is not possible to control precisely what goes out on the wire, and how fast the communication will be done. Packets might be lost and re-sent, different routes might be used, etc. Even if the transport is only done within a local network, different parameters in the transport still have influence on the speed.

Testing the system as a whole and defining throughput is even harder to do. Not only does the network setup influence the throughput, but overall performance will vary as a function of hardware, configuration of RAM and CPU speed, among others. The deployment of the system will also have a great influence on the throughput. Placing individual modules on different computers as shown in Figure 30

Worst case scenario of boundary communicationwill slow down the system. External network communication is much slower than internal communication where no network boundaries are crossed. For this reason, a precise number for the system can not be given, however similar repeated tests will give an indication of the range the test results will fall inside. If the system falls outside the predefined range, some kind of counter measures must be introduced in order to keep the system stable and running. The best approach would be to look at small subparts of the communication, and concentrate on optimizing those parts.

Given the fact that the system is depending heavily upon web services and hence SOAP, it is a logical step to look at the amount of data that is sent out on to the network. The size of the raw SOAP packages can be compared to different kinds of encodings, for instance, MTOM, ZIP compression and binary encoding. Encoding is the process of transforming a message into a sequence of bytes. Decoding is the reverse process. It is not possible to be sure on how much data is sent from a computer, unless the package is inspected right before it is sent out on the wire, and subsequently once it is received again.

A test is constructed where a custom filter is placed on the communication stream. This will count the number of bytes sent, and the number received. At the same time a timestamp will measure the time it takes for a message to have a round trip in the system. Different package sizes must be sent in order to have an indication of the impact on data size on different protocols.

The encoding measurements will indicate what the requirements for the network system will be. However, the time will also indicate the impact an extra applied filter will have on the system. For instance how much extra time will it take for a round trip if using encryption in the system? Will the compression make the overall system faster due to the smaller data amount needed to be transported on the wire? Etc.

WCF provides a way to place filters inside the communication stream, and hence monitor the data send and received. Of course the filter will add a little overhead on the system, but taken all the other sources for error, this is considered a minor issue.

To create a custom filter, three classes must be inherited for the encoding and another one if the class must be configurable from a configuration file.

Class	**Description**
BindingElementExtensionElement	Class returning a custom binding. Configuration of the class can be set through the configuration file, for example the encoding style of the binding, can be set when creating the *MessageEncoderFactory*
MessageEncoder	Base class for encoding messages. Contains methods *WriteMessage* and *ReadMessage*

Class	Description
	taking a stream and converting it to a message object and visa versa
MessageEncoderFactory	Abstract class that makes it possible to implement a factory object for the encoder. It is possible to have several *MessageEncoders* inside each other, but this can only be done in code.
MessageEncodingBindingElement	Binding element specifying the message version used to encode the message.

The custom encoder can be used for any insertion into the message transport, it counts the bytes sent on the wire, and at then creates a zip compression filter for the traffic. The compression can be turned either on or off. In the encoders *ReadMessage* and *WriteMessage*, a message is received and converted to a byte array. Compression is achieved by creating a *GZipStream*. The stream compresses the data using the GZIP industry standard available form RFC 1952. Because this is a standard available in different platforms, non-WCF clients can still grab the stream and decompress it using its native GZIP library. The stream takes an array of bytes and returns a compressed version.

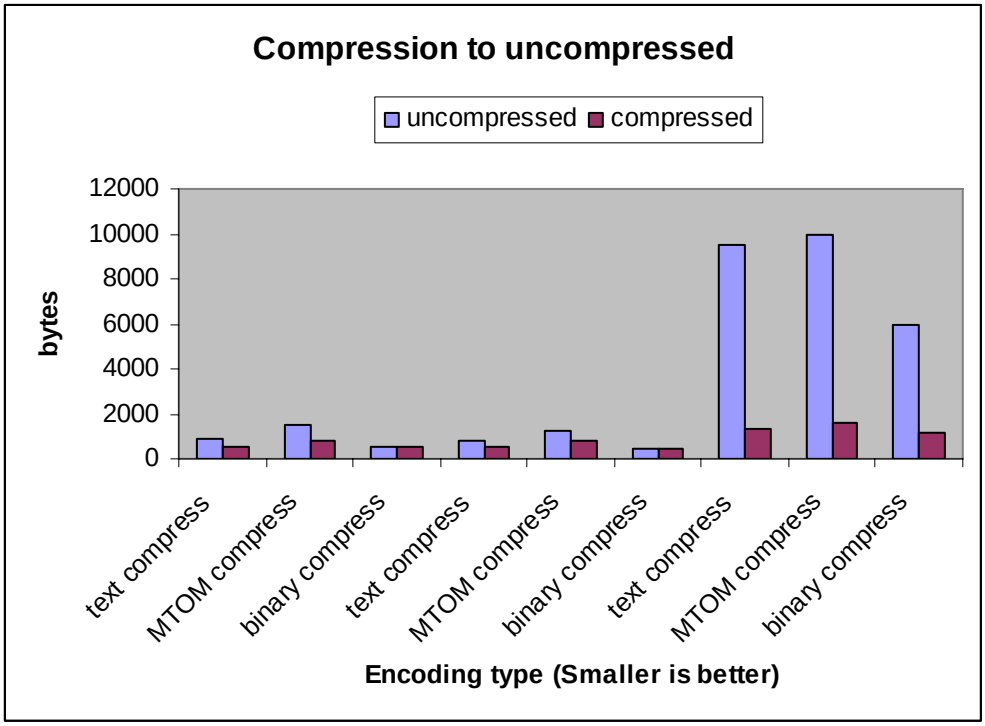


Figure 35
Compression of bytes

The test is run with 3 kinds of calls. The first service call is a simple one which retrieves the logical node directory from a logical device on the system. The server is not spending much time while creating a response, so the most of the time is spent on the web service traffic. The medium call issues a *logon* followed with a *logoff* service call.

Here the traffic sent is at a minimum level where the server processing time is slightly higher than the first call. The last example issues a *GetDataValues* service call where it can be regulated to send more or less data. Each method is called repeatedly for 10 seconds where the number of bytes and calls for each session is counted.

There is no doubt the compression can reduce the data amount sent on the line, however the price of this can be seen in the number of calls that the system is able to manage. For instance for the text call transferring 9kb is reduced by 87%, but with a reduction in numbers of calls by 8% If the bandwidth is limited this is definitely a thing that could be considered. However, it does reduce the overall flexibility of the system. Figure 36 shows the numbers of bytes sent during 3 different types of calls together with the compressed versions of the same service calls.

Christopher Kohlhoff and Robert Steel [CKRS2003] conducted a study on SOAP for high performance business applications. They conclude that if the goal is a reduced number of calls, then the speed of the compression is the predominant factor in a fast network, while it is the message size that matters most with a slow connection. Their tests also show that the system environment can make the results vary considerably.

Before making the decision of using compression, the amount of data sent and bandwidth requirements must be specified first, then proper tests will uncover if it is a good idea or not.

MTOM is intended for encoding binary data and do not provide any benefits on the contrary for plain-text encoding. MTOM could be used effectively in file transfers.

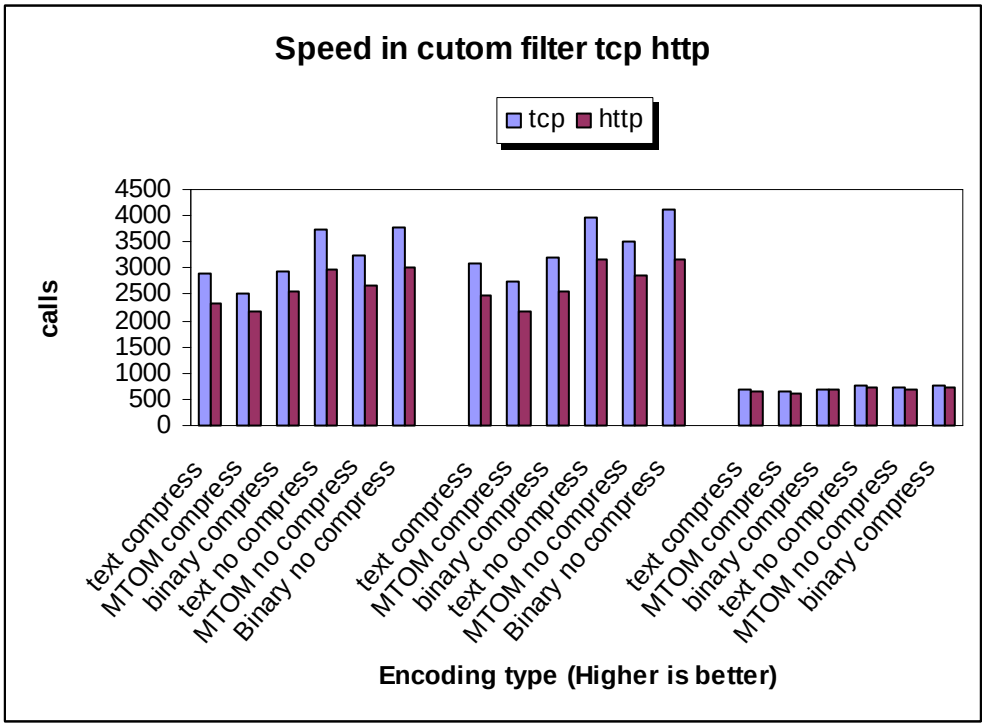


Figure 36

Bytes send for compressed and uncompressed messages by changing the transport protocol

On the network transport layer the transport can also be changed. Figure 38 shows how changing the transport protocol form HTTP to TCP effects the system by 3-20%. Please refer to the Appendix for the test results.

4.4.1 Speed test of different bindings

In this test the focus lies on choices made for the transport. The test will not focus on the speed on the system itself, but on choices made in order to effect the transport. Different transport protocols' security settings are changed where a test program call a predefined service method as many times as possible within 10 seconds.

Because the data sent and received per call is the same, it is possible to compare the results. However, it will only be an indication because the more time that the program spends inside a method, the less important the choice will be compared to the overall performance. To give a more precise result, different service methods are used for the test. The methods are rated according to how much work the server has to do to fulfill the request.

The table below shows the number of calls with different protocols and a comparison of the number of calls to the TCP protocol, the one able to complete most service calls[17].

[17] All tests are executed on the same computer, retrieving data from another computer in the same network. The test results should not be seen as an absolute speed of the system.

basic	3531	1693	1447	24,50%	41,17%	2,97%
basic mtom	3165	1794	1391	38,89%	33,22%	7,12%
pipe	4222	2345	1688	4,12%	1,92%	-11,73%
tcp	4396	2390	1490	0,00%	0,00%	0,00%
wshttp	1216	687	675	261,51%	247,89%	120,74%
wshttp mtom	1106	658	619	297,47%	263,22%	140,71%
wsCert	1414	732	699	210,89%	226,50%	113,16%

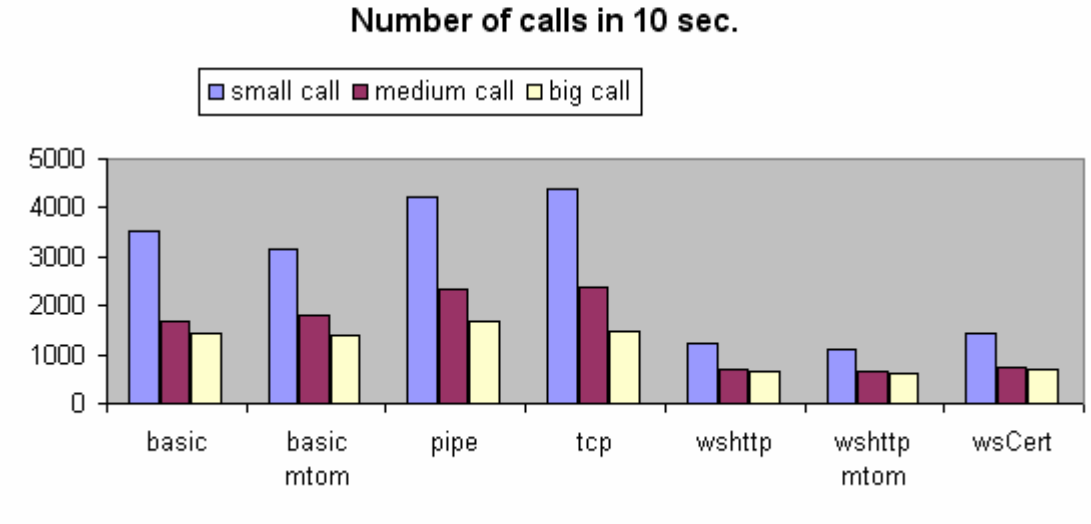


Figure 37
Speed test for different endpoints[18]

The test shows that depending on the protocol, the number of calls can decrease by up to 260% if *WsHttpBinding* is used compared to *TcpNetBinding*. This is remarkable because *WSHttpBinding* is the most flexible choice, but also one of the slowest!

The call revealing the high percentage is not sending very much data, but also the big call has over 110% better performance. It is worth noting that the last case uses mutual certification of the traffic. It has better performance than the basic WsHttp call. By default *WsHttpBinding* uses message security with windows login credentials.

Multi bindings test

The test has a client and a server. Both client and the server are configured through their individual configuration files. Each test process gets an endpoint for the service. The endpoint has the possibility to define every aspect of the communication, by defining behaviors for the endpoint.

[18] Small call GetServerDirectory
Medium call Logon Logoff
Big call GetdataValues for 7 tFCDA

The key attribute for the *appSettings* can be read from the application, and can be used for specifying the URI that the service is listening on. Each endpoint will be used by the client machine to run the same tests.

```
  <appSettings>
    <add key="WsHttp" value="http://host:8000/" />
    <add key="Tcp" value="net.tcp://host:8001/" />
    <add key="Pipe" value="net.pipe://host/" />
  </appSettings>

  <system.serviceModel>
    <services>
      <service
          name="IECServer.IEC61400_25Service">
        <!-- use base address provided by host -->
        <endpoint name="WsHttpBinding"
            address="HTTP"
            binding="wsHttpBinding"
            contract="IECServer.IECInterface" />

        <endpoint
            address="TCP"
            binding="netTcpBinding"
            contract="IECServer.IECInterface"
            name="NetTcpBinding" />
.
.
```

Figure 38
Multi endpoint setup in the configuration file

Please notice that the use of endpoints is an easy way to build a service where different client types are supposed to use it. For instance it is possible to have different encodings for different clients.

This test shows that what is transmitted impacts heavily on the round trip time of a call. The most important point in designing this web service is that it be capable of communicating with the end user. But if it is known from the start that the system will only be communicating with a WCF client, then there is no reason to keep using text as the encoding, or for that matter to only use the WS compatible protocols. *NetTcpBinding* would be much faster, and increase the overall throughput of the server.

Figure 39 shows how it is possible to choose a protocol out of some simple questions. To get the best performance and still have a flexible system, multiple endpoints exposes a great solution. A client should always look for the most optimal available endpoint in order to maximize performance and if the optimal solution is not possible get the next possible endpoint.

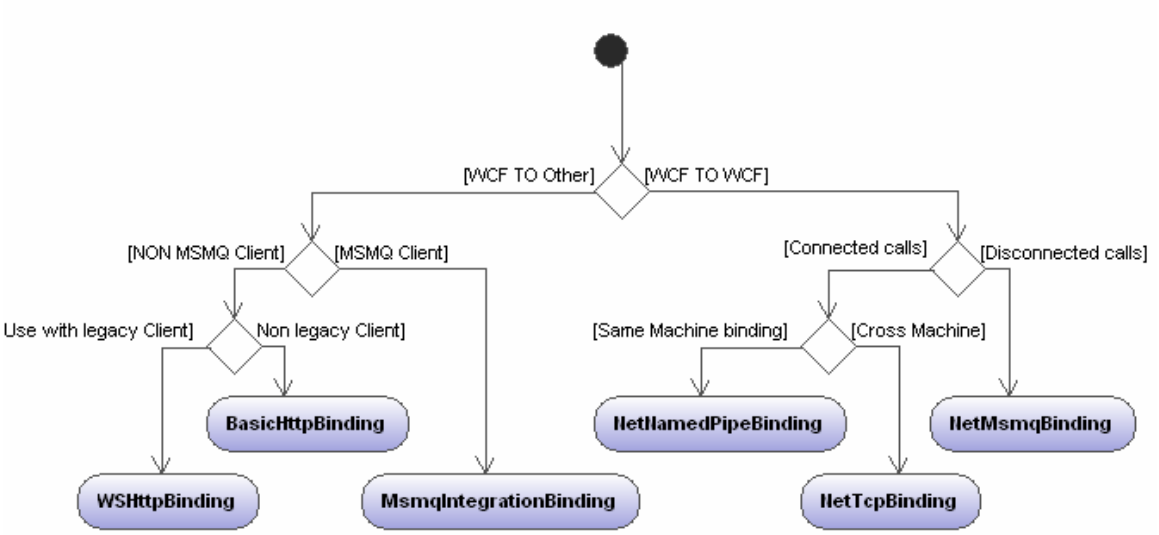


Figure 39
Binding choice

Figure 39 shows how the decision about binding protocol can be made. It is not suggested that every possible endpoint is exposed, all the implications that a binding entails must be considered. But an important point in this is that the design does not have to be centered on the lowest common denominator for all clients to be able to communicate with the service. Different endpoints can be created for different clients.

Case Summary

According to Bauyssounouse and Sifakis, Quality of Service is;

"*…function mapping [of] a given system instance with its full behaviour into some quantitative scale.*"

This is usually associated with the error rate, response time, bandwidth and system usage such as CPU and RAM peak. Usually quality of service (QoS) measurements depends upon both environment and service behaviour. For this reason it is hard to provide a clear number for what the test shows.

It should be noticed that the different standard encodings did have an influence upon both speed and data amount sent. It is hard to conclude something generally on the test. The configuration of the test and the system on which the test is running has so many parameters that many more tests could be conducted. For this to make any sense, a clearer goal for the system must be established. The test did show that the system reacted to different configuration parameters, for this reason it must be concluded that if the working condition of the system requires something special, the configuration does provide boundaries that can be tweaked for better performance for a special case, and thereby change the overall QoS.

No matter how the system is configured it will impact the overall operation of the system. In regular applications, the flexibility of the system was limited. However, due to the nature of web services and the way they are configures by using WCF, it is easy and cheap to provide several ways for clients to communicate with the system. No additional code is needed by just adding the endpoint to the configuration file. Not all clients support all possible choices by which the service can communicate. By exposing multiple endpoints each client can choose the most efficient one. This ensures that the system does not have to operate only under conditions designed for a worst case scenario, but it can provide different levels of service. This maximises flexibility for both client and service providers.

4.5 Case Study 4

Part 5 of the IEC61400-25 series is dedicated to conformance testing. Lots of tests are involved when developing a complete system until it is running properly and satisfying the specified requirements. Testing a complete system involves several kinds of tests. Some of the tests can be completed during the development process, while others must be performed upon completion of the system. The completed system should be tested successfully against all the proposed tests. What to test for varies a greatly but the most common attributes a system should have include capability, reliability, efficiency, portability, maintainability, compatibility and usability. This thesis is about a prototype, and as such it is not intended as a complete system. The prototype is mainly to test aspects of a complete system. In order to create a final system more formal requirements must be specified, and the system should be designed in order to meet those specific requirements.

Part 5 proposes various test procedures in order to make sure the system operates properly according to the IEC61400-25 standard. However, all the procedures involved in the conformance tests are not conducted under the development cycle of this project. This would be far too unrealistic with the limited resources available.

The tests conducted for the developed prototype are limited to the functional testing of the services offered and general unit testing during development of the prototype. The aim of functional testing was to test whether the service methods operated as they should with proper request and reply objects.

Functional tests applied on the service methods also reflect the inner state of the server which makes it possible to trace behaviors of the system under certain conditions which will lead to further structural testing procedures. However the main focus is on functional testing.

In Part 5 of the IEC61400-25 series abstract test cases are defined to test a server. The structure of the server test cases is as follows:

a) Documentation and version control
b) Data model (IEC61400-25-2)
c) Mapping of ACSI models and services (IEC61400-25-3); the corresponding sub clauses that define the abstract test cases are listed below:
    - Application association.
    - Server, logical device, logical node, and data model.
    - Data set.
    - Reporting.
    - Logging.
    - Control.
    - Time and time synchronization.

The tests conducted for the prototype mostly covers part b) and c) of the list above. The service methods through which the tests are conducted are listed below.

- Logon
- Logoff
- GetServerDirectory
- GetLogicalDeviceDirectory
- GetLogicalNodeDirectory
- GetAllDataValues (Not implemented)
- GetDataValues
- SetDataValues
- GetDataDirectory
- GetDataDefinition
- GetDataSetValues
- SetDataSetValues
- CreateDataSetValues
- DeleteDataSetValues
- GetDataSetDirectory
- Report
- AddSubscription
- RemoveSubscription
- GetBRCBValues
- SetBRCBValues
- GetURCBValues
- SetURCBValues
- GetLCBValues
- SetLCBValues
- QueryLogByTime (not implemented)
- QueryLogAfter (not implemented)
- GetLogStatusValues (not implemented)

Each test clause is divided into two groups as Positive and Negative test cases. Tests grouped under Positive are test cases where the result is successful whereas negative test cases results in non successful responses. Please refer to the appendix for the relevant test results.

The service tests conducted show that the implemented service methods can generate the expected responses with the correct data types defined for the services. This implies that the business logic executed on the server is correct as well.

Chapter Summary

In this chapter different case studies have been conducted. This is done to make sure that the system meets the requirements defined in Chapter 2.

In the system a wide range of things have been addressed, and a proposal has been presented. It is hard to say if the choices will meet requirements in a real system, unless more formal requirements have been defined, however the case studies do show preliminary success at the conceptual level.

Chapter 5

CONCLUSION AND FUTURE WORK

The goal of this thesis was to implement and test an IEC 61400-25 compliant generic server which will be used to monitor and control wind power plants. During the development of the server there has been no opportunity to test and develop against a real wind power plant device. Instead of working against a real device, a basic device simulator has been created to create a minimum environment to develop a prototype and test it.

The evaluation of the standard is made through the degree on how successful the developed system is working. That is, only the developed communication system has been considered as a base when evaluating the standard. Specific details within the general wind power plant domain are not considered since that is beyond the scope of this thesis. For example, the real behavior, constraints and limitations of a WPP are not reflected in this study.

The main focus has been on the server system and its interaction with clients through the specified web services defined in IEC 61400-25-4. When developing the server system, general considerations such as security and modularized software architecture have been taken into account and implemented, wherever applicable.

From the requirements outlined in chapter 2, a prototype has been implemented and four case studies have been conducted in order to verify that the system meets the specified requirements.

5.1 Chapter summery

Chapter 1
Chapter 1 presents the motivation for having a platform neutral communication and control system for wind power plants. At the same time it argues that such an approach has value in order to communicate without vendor boundaries. It is shown that this is one of the main building blocks towards a large scale monitoring and control of the power grid.

Chapter 2
Chapter 2 presented the architecture of the system and the data model that the system should be in conformance with together with a conceptual model defining the modules constituting the system. Through the analysis it was specified that the system must provide the following features:

- It must represent the wind power plant information model as defined in IEC 61400-25-2
- It must be comprised of replaceable modules.
- It must exhibit secure web service communication between client-server and the different modules the prototype is comprised of.
- It should be an easy to configure, modular architecture.
- It should have the capability to handle data in a way that ensures maximum throughput.

.

Chapter 3
Chapter 3 presents the arguments for using SOA and web services for the prototype implementation. Then a description of what can be expected from the use of today's standard web services is presented followed by a discussion on how it is handled by Windows Communication Foundation (WCF). WCF was chosen a development platform using .NET C#.

Chapter 4
Chapter 4 presents case studies, where it is demonstrated that the prototype system is capable to work under different conditions. This chapter also describes how the involved modules are implemented together with their capabilities.

5.2 Conclusion

The goal was to implement a functioning prototype of a generic server. However, there are still many aspects that must be dealt with in order to say that everything has been implemented. Almost all the services defined in the standard have been implemented using web services. This includes the mapping and representation of the data on the server side and the communication of the data to client systems.

The IEC 61400-25 standardization series is still work in progress, and during this project several updates have been released. The new releases of the standard were mainly on ACSI service mappings to a communication profile. This is part-4 of the series defining the web service interface with its data types. After some time this project had to be closed for changes in order to continue the implementation from a fixed point. Otherwise, continuously modifying the system would slow down the implementation. Because of this, not all changes made to the communication mapping are reflected in the implementation. However, services implemented earlier have been changed to conformance with the later versions of the mappings.

Examples of services implemented according to the older versions are the *Logon* and *Logoff* services used to authenticate and associate a client system. In later versions of the mapping these services have been replaced with similar services making it possible for a client to participate in more than one client-server association. Currently the server only recognizes the Logon and Logoff services when associating with a client system.

Some of the services are not implemented because their WSDL definition was not complete or erroneous. One example to these services is the *GetAllDataValues* which was not associated with a suitable return type.

The list below outlines the main requirements which have been implemented in this project.

- **Information Model:** On the main server a suitable data structure has been established which can store data as defined in IEC 61400-25-2. It is an in-memory data structure making data manipulation and retrieval very fast and efficient. When building logical devices from logical nodes and data instances, the exact hierarchical structure can be presented on the server. The information model on the server is initialized through an XML file which makes it possible to alter the data models of the different devices easily.

- **Service interface:** A web service interface has been implemented through which the service methods are communicated. The web service is implemented from the WSDL given in IEC 61400-25-4. WCF has been used for the web service in order to benefit from latest improvements within web services.

    - Different endpoints providing different communication features can be set up for the service so that client systems can choose the best suitable endpoint. Tests in case study 3 shows that different choices regarding the communication (encoding, encryption, transport protocol) will result in big performance variations.

- **Services:** The server logic for the services has been implemented. Business logic within the different modules has been isolated from the server. It is possible to send service requests with the correct data types defined in the standard and retrieve the corresponding response.

- **Module based architecture:** The server components have been implemented as separate modules making it easy to configure and change the system. For example the service interface is not tightly coupled with the central server itself.

    - It is possible to attach another service interface using another protocol than SOAP/web services.
    - The change of the AC module is only a matter of replacing a DLL in a directory. It is not even necessary to provide any configuration changes, the system will automatically recognize that a module containing AC information is present, and it will automatically be loaded.

- **Reporting:** The reporting service has been implemented according to the event driven data retrieval model defined in the standard. A client system can subscribe to reports which are generated by the report control blocks initialized by the

subscription service. Both buffered and un-buffered reporting has been implemented.

- **Access Control:** An access control mechanism is implemented making it possible to create a specific view of the data and services applied to it. The access control module can be configured so that different views of the system can be assigned to different client systems.

- **Device simulators/connectors:** In order to be able to test the system, device simulators capable of generating data have been implemented. The simulators use log files from real wind power plants when generating data. Different scenarios have been studied which are most likely to appear in a real world case such as connection loss and lost data packages.

- **Configuration:** The server and its modules are implemented such that they can be easily configured. For example the access control mechanism can be configured through XML files in order to create different views of the system.

    - The server can be configured to handle different WPP simulators with different data models. The structure of each simulator attached to the system can be defined separately.
    - The device connectors can be configured so that the server can locate the simulators and associate it with the respective data model defined on the server.
    - The security and transport features of the system can be easily configured through configuration files.

- **Client:** A web based client has been developed to test and demonstrate the services implemented. The implemented communication module for the server is based on web services. Any client capable of consuming web services can connect to the server and perform its tasks through the web services made available.

5.3 Future work

At the end of this thesis project, all work is far from done. The system contains many aspects that would benefit from continued work. During this thesis project the standard has not been finalized, though it is expected to be so in the near future. When this is done a complete implementation can be realized. The server must be tested in accordance to the complete standard and some of the methods must be re-implemented.

- **Logging:** The implementation of the logging features of the server is not fully completed. It is only possible to configure the log control blocks (LCB) and retrieve their data directory. However, the logging is very much similar to the reporting mechanism. Both mechanisms make use of a handler object which is controlled by the control blocks which works in conjunction with an event monitor. Since the

reporting mechanism is working, it will not be unrealistic to conclude that the logging mechanism can be implemented easily by reusing some of the important techniques used in the reporting mechanism. However, one important difference in the logging mechanism is the log database. Instead of short term buffers the logging must be implemented together with a relational database so that clients can retrieve historical data which has been logged. The service methods to retrieve log data have not been implemented.

- **Test in real life:** From the WPP to the client, the data has to be passed through many levels of transportation, each level placed in separate modules or execution threads. The fact that the system is split into so many modules provides flexibility, but having many communication lines will slow the system, and at the same time expand the list of places where something can go wrong. Future work should include a study into what is needed for each boundary cross, both in relation to security, stability and performance. The prototype provides a proof of concept showing that is possible to implement the standard and have a running system. The next step should include a real case study, where a scenario is designed, and the system is employed for a real wind power plant. With these requirements a more formal test could be conducted evaluating the performance of the system. This should lead to optimization of the individual modules in order to have the best throughput for that particular case.

- **Security aspects:** Security must be looked upon in depth before the system can be accepted as a completed final system. The focus has been on implementing as much of the standard as possible. In the communication a lot of boundaries are crossed. In order to have a user-stable and friendly system, more work should be put into analyzing the possible failures that each module can experience, and a common model for distributing information about other module states could be designed.

- **Stability and robustness**: Currently the server maintains an in-memory internal data model for holding the device data. The data and overall state of the devices are not stored for later recovery, including reports stored in internal buffers. The server should implement features that make it possible to recover from an unexpected failure.

5.4 Overall Conclusion

The main focus was on creating a system to implement a server system compliant with IEC 61400-25 standardization series, a system that was not tightly bound to specific technologies and easy to configure. The resulting implemented server system has shown that such a system is possible.

The system covers most of the major parts defined in the standard. All topics in the requirements has been addressed one way or the other, and we believe this thesis

should provide a good basis for future work and efforts toward the goal of creating a vendor independent communication environment for all wind power plants in the world.

APPENDIX

A TESTING

TESTCASE

Testing a complete system involves several kinds of tests. Some of the test can be completed along the way while development process is in progress, while other requires that the system as a whole is completed. For the system to be completed all of these test should be run and passed. What to test for varies a great deal but the most common attributes a system should have includes capability, reliability, efficiency, portability, maintainability, compatibility and usability.

This thesis is about a prototype, and as such it is not intended as a complete system. The prototype is mainly to test aspects for a complete system. In order to create a final system more formal requirements must be specified, and the system should be designed in order to meet those specific requirements.
A common conception about test is that they should reveal errors; however, more recent thinking is adopting the notion that a good test is one which reveals information of interest to someone within the project.

Unit test

Unit test is small automated test that the system can run on demand. As the name implies a unit test is suppose to test one unit in the system, or one small operation. Unit test is usually combined in test groups called sockets.
The tests are created by specifying methods, and associate the [test] attribute with it.
The some code set up the test the data there is to be tested. In this case it is a test for testing the conversion of the data between the server class and the WCF class called `ConvertToWCF`.

The initial object is created and the method is called. The reply object is the tested to make sure that it contains what is expected of it.

The assert object contains many method for testing the result. In this case it looks for equality in the reply.

```
[Test]
public void AddSubscriptionReply() {
      //create test object
      AddSubscriptionReply initWCFObject =
      new AddSubscriptionReply();
      //fill object with data
      .
      .
      //call method
      AddSubscriptionReply ReturnObjServer =
      AddSubscriptionReply.ConvertToWCF(initWCFObject);
      .
      .,
      //check respone
      Assert.AreEqual(ReturnObjServer.Items[0], setData);
}
```

Figure 40
Unit test setup (some code has been removed)

When running the unit test, reflection is used to call all methods containing a test attribute. Each test must be completed without any error in order to be a success. If all tests are completed without errors the test is a success, else an error indicating the problem will be displayed.

What unit test can do and not do.

The great advantage of unit test is that if a method is altered the unit test might identify errors introduced in the change. Unit test do not replace common sense. It is not an automated test. Unit test is not any better then the programmer who wrote them.
They can tell if the system works correctly, but only is an operation within the system is working correctly. Of cause having as many of the operations covered by unit test as possible will provide a good indication if the system is working or not. The test can not be used for testing internal states or operations, they simply chain a specific input together with an expected output, for instance if a test on *setDataValue* is performed, the test will not reveal if this has been done or not, this kind of testing is known as black box testing. In order to check the internal state the data must be retrieved by *GetDataValue*, this is known as white boxing test.

Unit test should only be used for non time critical tests, and not to test performance issues of the system.

Acceptance testing

The purpose of an acceptance test is to make sure that the system meets the requirements that are specified in the requirements specifications. This system is a prototype and for that reason there are no formal The prototype has not been subdued to acceptance test because no formal requirements has been presented for the final system.

User Testing

User test is not relevant for the main scope of this project. A user test is conducted in order to establish if the system is usable for the end user. Client programs should be submitted to user test.

Additional possible test

In Visual Studio Team Suite it is possible to get an indication of how much of the code was covered by the unit test, it is also possible to obtain information about the time the running application spend inside different part of the program. This is a great tool for optimizing code and get the most out of the time spend in tweaking the system. If the system spends a lot of time inside a special subpart of the program this would properly be a good place to look for optimizations in the code. In order to run these tests the system must be deployed and running under normal circumstances. No such test has been performed, because no data about normal circumstances is available, and the test computers do not have Visual Studio Team Suite installed on them.

APPENDIX

B CERTIFICATE

Access to the WPPs data must be restricted to approved parties only. Mutual knowledge of identity must be established, in order to removing a possible traffic relay to another server (relay or man-in-the-middle attack) or people masquerading as a legitimate client

Both the service and client must have there identity verified by the other party, one way to do this is by the use of a certificate. When the client contacts the server, certificate information is exchanged together with a mutual encryption key. Both client and the service must be able to verify the claim of identity for the communication to take place. The most common ways of this is if the certificate is signed by a certification authority, or the certificate is in a trusted key store of the machine checking the claim. If the certificate is signed by a root authority or it is located in the key store the certificate is trusted.

Getting a certificate signed commercially can be a costly affair. In a closed community like a corporation where a finite number of users exist, there is no reason to have a commercial authority to sign the certificates. A root cert is created and placed on the server. Each user gets a certificate signed by the root cert. When a user presents a claim of identity it can be verified by the signature.

For the test 3 certificates has been created for the system

- Root cert
- IECServerCert (named localhost)
- IECClient

The cert is placed on the local computers proper key store make sure that the process running the applications has appropriate access rights to the files the cert is located in. On an IIS (Internet information server) this is the standard the user ASPNET, and the rights can be set by using the utility *certkeyfiletool.exe* distributed together with IIS.

In the servers configuration file the following lines must be included.

```
<endpoint address=""
      binding="wsHttpBinding"
      bindingConfiguration="Binding1"
      contract="IECServer.IECWCFInterface"/>
.
.
<binding name="Binding1">
   <security mode="Message">
      <message clientCredentialType="Certificate"/>
   </security>
</binding>
.
.
<behavior name="IECServiceBehavior" returnUnknownExceptionsAsFaults="False">
<serviceCredentials>
   <serviceCertificate
            findValue="IECService"
            storeLocation="LocalMachine" storeName="My"
            x509FindType="FindBySubjectName" />
      <clientCertificate>
        <authentication
            certificateValidationMode="ChainTrust"/>
      </clientCertificate>
   </serviceCredentials>
</behavior>
```

Figure 41
Service setup for certificate.

Inside the security tag it is specified that the security is message based opposed to transport based, and the client must present a valid certificate as credientials. Credientials can also be established as Basic[19], Digest[20], windows[21], NTLM[22], but certificates are based on open standards, and are therefore the most flexible choice. The certificateValidationMode is set to ChainTrust. Telling the computer that the cert must have been signed by a trusted authrity.

19 This corresponds to the Basic authentication method in IIS. When using this mode, the IIS server must be configured with Windows user accounts and appropriate NTFS permissions.
20 Digest authentication is similar to Basic authentication, but offers the advantage of sending the credentials as a hash, instead of in clear text.
21 This corresponds to Integrated Windows Authentication in IIS. When set to this value, the server is also expected to exist on a Windows domain that uses Kerberos as its domain controller. If the server is not on a Kerberos-backed domain, or if the Kerberos system fails, you can use the NTLM value below.
22 This allows the server to use NTLM for authentication if Kerberos fails. For more information about configuring IIS in IIS 6.0

```
<endpoint name=""
address="http://localhost/cert/service.svc"
binding="wsHttpBinding"
bindingConfiguration="Binding1"
behaviorConfiguration="ClientCertificateBehavior"
contract="IECServer.IECWCFInterface " />
.
.
<binding name="Binding1">
   <security mode ="Message">
      <message clientCredentialType="Certificate" />
   </security>
</binding>
.
.
<behavior name="ClientCertificateBehavior">
   <clientCredentials>
      <clientCertificate findValue="IECClient"
            storeLocation="CurrentUser" storeName="My"
            x509FindType="FindBySubjectName" />
         <serviceCertificate>
            <authentication
      certificateValidationMode="PeerOrChainTrust"/>
      </serviceCertificate>
   </clientCredentials>
</behavior>
```

Figure 42
Client setup for certificate

The client is set up in the same way as the service. Please notice that the *certificatValidationMode* is set to *PeerOrChainTrust*. The system will look for the cert in the local key store of the current user; if it is not found here it will look for the signers' certificate. Having the peer to check is not as safe as checking against a signer, because anyone can in theory add a certificate to a key store, but it is not everyone that can sign the certificate with the right key. The use of certificates fulfills the requirements of security for case study 1. It takes some efforts to set up the system, but it should still be worth the effort by the extra security added.

The certificate assures the traffic between the web service client and the web service itself is secured. If the communication stub is shared by many users, the identification of them must be established elsewhere. If the client is a DLL used by a web server it will not identify the end user, to do this a simpler security approach must be taken between the web client and the server.

Running the demo

Even though the use of certificates for security is a great thing for the system, it is not that easy to setup a system that runs on any system. Information in the certificate must correspond to that of the host machine, and be proper installed into the host systems

trusted key-store. In order to run the demo this must be done, otherwise the system will not be able to run properly.

APPENDIX

C VERSIONING AND WEB SERVICES

The lifecycle of a system contains maintenance, in which the system is submitted to modification and/or additions. To make a flexible system it is vital that changes can be introduced in an easy way where the possibility of introducing errors is limited to minimum. A common challenge for software designers is how to add or change functionality with out resulting into much conflict and still be working for the running system. Changes might include change in data types or functionality.

For regular applications, where the whole code is developed and controlled the same company, changes can be enforced, and every system using the module must be adapted to the new convention. With web services this is not so easy. It might not even be known how many users a service has. Changing the WSDL does not necessarily result in the client not being able to use the service; however it can result in unexpected replies.
Changes can be either backward-compatible or non-backward-compatible.

Backward compatible changes include:

- Addition of new operations to an exciting WSDL,
- Addition of new schema types in the WSDL.

Non backward compatible changes:

- Removal of an operation
- Renaming of an operation
- Changes of parameters including data types and order for an operation
- Changes of the structure of a complex data types.

Backward compatible changes can be implemented without that many restrictions. Running services will not be affected by changes because they still hold a valid communication contract for the service.

To be backward compatible the involved data types and method names must not change. It is always possible to add data or methods, but not to omit them entirely. No backward compatible contract need to have a way for the client to determine if a change has been made.

In web services there is no standard way determining the method version. This leaves the problem up to the developer to solve the problem through patterns and best practice schemas. A common way to deal with this problem is to have a version number assigned for a method. In this way different versions can be supported depending of the version number a client is able to use.

In any case it would be a good idea to have a policy on the field. This will inform the clients about what to expect for changes. In that way the clients has the possibility to design there client system in a way where versioning will not result in major problems. [KBME2004]

APPENDIX

D REFERENCES

[JDAT03] José Delgado, President & CEO American Transmission Co.
The Blackout of 2003 and its Connection to Open Access

[IEC] IEC 61400-25 Communications for monitoring and control of wind power plants

[IBM04] SERVICE-ORIENTED ARCHITECTURE AND WEB SERVICES: CREATING FLEXIBLEENTERPRISESFOR A CHANGINGWORLD October 2004 Ziff Davis Media Custom Publishing.

[Infoworld] http://www.infoworld.com/article/05/11/03/HNjavanet_1.html?WEB%20SERVICES%20INFRASTRUCTURE

[CKRS2003] Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems http://www2003.org/cdrom/papers/alternate/P872/p872-kohlhoff.html

[WY2005] Improve the interoperability between J2EE and .NET, Part 2. wangming Ye Software engineer IBM 2005. http://www-128.ibm.com/developerworks/java/library/ws-tip-j2eenet2.html

[WSI] http://www.ws-i.org/

[BRJ99] Booch, Grady, James Rumbaugh, and Ivar Jacobson . Unified Modeling Language User Guide. Addision-Wesley, 1999.

[Fow02] Fowler, Martin . Patterns of Enterprise Application Architecture. Addision-Wesley, 2002.

[BCK03] Bass, Clements, Kazman; Software Architecture in Practice (2nd edition), Addison-Wesley 2003.

[WY05] IBM article Wangming Ye
http://www.128.ibm.com/developerworks/iava/librarv/author#author

[TPC06] Tpc website: http://www.tpc.org/tpcc/faq.asp

[KBS04] Dirk Krafzig, Karl Banke, Dirk Slama, Enterprise SOA: Service-Oriented Architecture Best Practices, Prentice Hall PTR 2004, 0-13-146575-9-12

[CKRS2003] Christopher Kohlhoff, Robert Steel Evaluating SOAP for High Performance Business Applications: real-time Trading Systems. 2003

[PY03]Prasad Yendluri 2003
http://www.webpronews.com/webdevelopment/webapplications/wpn-27-20030829WebServicesReliableMessaging.html

[JP2005]Joel Pobar 2005 MSDN magazine and
http://msdn.microsoft.com/msdnmag/issues/05/07/Reflection/

[MFPEAA] Patterns of Enterprise Application Architecture.2002

[KB2006] Keith Brown security Brifes
http://msdn.microsoft.com/msdnmag/issues/06/08/SecurityBriefs/default.aspx#S3

[KBME2004] Kyle Brown Michael Ellis Best pratices for web services versioning
http://www-128.ibm.com/developerworks/webservices/library/ws-version/

[FC] Frances Cleveland IEC TC57 Security Standards for the Power System's Information Infrastructure – Beyond Simple Encryption

APPENDIX

E BOOKLIST AND SITES

Name	Author	publisher	ISBN
Programming INDIGO	David Pollmann	Microsoft Press	Library of Congress Control Number 2005925788
Programming Microsoft Windows with c#	Charles Petzold	Microsoft Press 2002	0-7356-1370-2
Security in computing	Charles P. Pfleeger, Shari Lwrence Pfleeger	Prentice Hall 2003	0-13-035548
Patterns of Enterprise application architecture	Martin fowlser Matt Foemmel, Edward Hieatt, Robert Mee, and Randy Stafford	Addison-Wesley Professional 2002	0321127420
NUnits	Bill Hamilton	O'Reley Media	0-596-00739-6
Special edition using c#	various	2002 que	0-7897-2575-4
Enterprise SOA: Service-Oriented Architecture Best Practices	Dirk Krafzig, Karl Banke, Dirk Slama	Prentice Hall PTR 2004	0-13-146575-9

Site
In designing this project a lost of information was gardened from various sites concerning demos and knowledge.

Forums and blocks was used a lost. It is not possible to credit every single forum writer, but the most used sites are listed here.

http://msdn2.microsoft.com/en-us/default.aspx
www.Codeproject.com
http://blogs.msdn.com/drnick/
http://staff.newtelligence.net/clemensv/
http://www.douglasp.com/blog/default.aspx
http://blogs.msdn.com/yassers/
http://blogs.msdn.com/mfussell/
http://wcf.netfx3.com/content/BreakingChangesbetweenVistaBeta2andJuneCTP.aspx

APPENDIX

F GLOSSARY

Description of terminologies used

In this section a brief description of some of the used technologies, and other terms

Web services

A web service exposes an interface, and communicates through data types shared in contracts. The communication in and out of the service is done in a common industry agreed way, by using numerous XML protocols.

XML

XML is a self-describing, human readable language that is used to structure information. Xml documents are written in ASCII[23] characters. XML is simply a way to format a document. Xml is not at tightly bound language, anyone can define their own data types.

Because XML is based upon ASCII it is platform independent. At the same time it is a text format, assuring that all standard protocols for exchanging text can be used. No special measurement has to be taken when using XML. It can be passed through firewalls without any problems, and there is no chance of getting malicious code, because it is only comprised of text information. To use a XML document on simply has to follow the rules defined in a schema.

SOAP (Simple Object Application Protocol)

Soap is a schema for XML. It defines a way to send a message over a network. In environments where services are distributed among computers on a network, it is essential to have good communication. Earlier on the communication was offend done in a system specific way. For instance using RPC [24] or DCOM [25] the client were bound to these technologies not only might be bound to a specific kind of system, but it also required that the network was equipped for that kind of communication. For instance using RPC special port had to be open in order to fulfill the requests. SOAP utilizes standard web protocols like HTTP, which is supported by all browsers. SOAP is completely platform and language independent of the application using the service.

WSDL (Web Service Definition Language)

This is an XML format used when publishing a web service. This file describe all the basic essentials to communicate with a given web service. The file acts as a contract of how the communication has to be done in order to use the service.

23 ASCII American Standard Code for Information Interchange. A Common schema for representing characters on computer systems. Using ASCII makes sure that most computer systems understand the text.
24 Remote procedure call. Method to allow a programmer to call function located on a remote machine. Even though most people relate RPC to Suns variant, it is a pattern for solving distributed method calls.
25 Distributed Component Object Model (DCOM). An extension of Microsoft's Com objects, DCOM makes it possible to utilize COM object across a network. COM is earlier MS attempt of making language independent object for programmers.

A WSDL file contains four major elements.

- <porttype> :This describe the Operations there can be performed on a web service, and how the messages is used to achieve this. The port type can be compared to an interface for a class or function library in conventional programming.
- <message> :This is the signatures of the data entering and leaving the methods. It describes the data elements entering and leaving a method. Each method can have one or more messages attached to it.
- <types> : describes the data types involved in the communication. To have as much platform neutrality as possible XML Schema is used for defining the data types.
- <binding> : Defines the message format and used protocols.

Given a WSDL file you have an abstract description of the operations and messages involved, and a concrete binding to a URL and message format. WSDL files do have information about the functionality of the service, but in today's use of web services, there should be added a layer of security, and flexibility. This is addressed later when we talk about MS way of dealing with web services.

XSD

A XML format defining the rules/data types a given XML file can or must contain. The WSDL is defined using the XSD standards.

UDDI (Universal Description, Discovery and Integration)

What good is it to have a brilliant service, if no one knows that it is there? UDDI is a standard way to publish information about a web service on the internet. By registering to a UDDI service you make sure that people can find your service, and get the WSDL information necessary to consume it. Having a service like this is a cornerstone in having agile software, because several of the same type of service can be registered with a UDDI. This means that if one of the services has a denial of service, the other one can take over.

W3C

W3C is one of the corner stones in web standards. They play a major role in the standardization of protocol used for web services. This includes XML, WSDL, SOAP, MTOM and the WS-*.

OASIS

Oasis is one of the major forces behind web service specifications, this include Ws-security SAML. They are also involved in UDDI and WS-ReliabeExchange.

WS-I

WS-I group for open industry standard promoting web services and the interoperability Ws-I has released the most widely adapted web service profiles like WS-BasicProfile. They are also behind numerous tools for conformance testing..

Encryption

Encryption comes from the Greek word Kryptos that means hidden or concealed Encryption has the purpose to make information unreadable to anyone else then the intended parties of the information. This is done by some kind of key there can be used for unlocking the true meaning of the text. A good encryption algorithm has the following properties:

- Easy and fast to convert plain text into cipher text. (Encrypt)
- Easy and fast to convert cipher text in to plain text. (Decrypt)
- Guessing the right key should be close to impossible in linearly time[26]
- The cipher text must not contain hidden information about the plaintext[27]. It must hide information allover the cipher text.

In computer science there exist two major encryption methods commonly used today, synchronic and asynchrony encryption.
Synchronic encryption uses the same key for both encryption and decryption. These are usually the fastest encryption algorithms, but it has the big problem of sharing a common key between parties without the key getting falling into the wrong hands.

Asynchrony encryption has a public key and a private key. (Two large prime numbers). These keys are interchangeable and can decipher anything locked by its counterpart. Anyone can encrypt a text using the public key, and only the owner of the private key can read it. In the same way the private key can be used to encrypt or sign a text.
By using the public key the text can be decrypted.
Signing is done by calculates a unique number for the text, a hash value, and encrypt it. The client calculates the hash values, and the encrypted number must be equal to this.) Usually the private key will not be used for encryption, because everyone can read the text anyway, but signing is very useful. The signature does not only verify the identity of a message, it can also guarantee the integrity of it. If someone changes the massage, the hash value will not match, and the receiver will know that the data has been tamper with. At the same time everyone is guarantied that it is indeed the person who claims to have signed the document there has done it, because a wrong private key would make the text unreadable by the public key. Asynchrony encryption takes advantage of the fact that it is hard to factor large numbers.

[26] |Brute force attack where the attacker tries to guess the key by trying every possible combination possible.
[27] If the cipher is a rotation cipher (substitution of letters) vital statistical information is still hidden in the text about relation to other letters and frequency of use of each letter. For instance the letter E in the English language is used much more frequently then the letter x, Statistical analyses could then give a good idea of the substation letter.

Certificate

A certificate holds a collection of data there identifies a computer or a user. Everyone can create and distribute certificates, and for this reason create a certificate with whatever information they would like. If the certificate is to be useful, it must be possible to determine if the identification is from someone that can be trusted or not.
A certificate contains the following information.

- **Public key information:** This key is used by anyone who wants to communicate with the owner of the cert.
- **Certificate serial number:** GUID there identifies the certificate uniquely.
- **Certificate period of validity:** When creating a certificate it must be specified for how long it is valid. After the date the cert must be either removed or resigned, in order to still prove the identity
- **Server host address**: The certificate must belong to the domain that the client is trying to contact.
- **Name of the authority who has certified the certificate. Authorities name or domain:** The client checks if the signer is a trusted authority. Every computer has a list of trusted CA and there public keys.

The Certificate Authority's signature

It can be confirmed that the certificate is valid by using the public key of the Cert authority to check that he is who he says he is. For this to work the Certificate authorities must be in the list of trusted authorities on the computer.

The certificate holds an asynchrony key pair, there and be used for encryption of data.

A common way to distribute certificates is through the x.509 this is a common way to distribute certificates

APPENDIX

G JAVA FRAMEWORKS FOR WEB SERVICES

JAVA initiative for supporting WS standards

In this thesis Microsoft products has been used, but it is vital that the service can be used in other environments.

WCF is built upon WS-* standards and oasis standards. It is a new programming model, with a lot of new features, so the question is if it is possible to create a client by for instance using the Java technology. For the service to be consume the WSDL, there are several things the java client must understand. Java do not in it self have good support of the new web services, but several projects are in the making which will make it easy to combine Java and WCF. In late 2005 Ashesh Badani, group marketing manager for SOA at Sun announced that java will support many of the WS-* standards.[28] [InfoWorld]
Two of the most interesting ones are:

- Wsit (Web service interoperability technology): It is a code base enabling interoperability between the Java platform and WCF. It is distributed under common development and distribution license.[29]. The project is still in its infancy, and is for the time being not documented that well or tested. A demonstration at the javaONE conference in San Francisco showed a functioning java -WCF connection. More info can be found at: http://www.opensource.org/licenses/cddl1.php

- SCA (Service Component Architecture[30]). This is the java answer to WCF. It provides an environment for building web services in java. SCA like WCF rely heavily on WS.* specifications. SCA can also be used from c++ and other similar languages.

It must be concluded that WCF do not limit the reuse in other environments, like java, as long as only common profiles are used.
This is also shown incase study 1 where AXIS are used for connection to the service.

28 The specifications include:WS-Addressing, MTOM, * WS-Policy, * WS-MetadataExchange ,* WS-Security ,* WS-Trust ,* WS-SecureConversation ,* WS-ReliableMessaging ,* WS-Coordination
[29]CDDL http://www.opensource.org/licenses/cddl1.php
[30] Created by IBM, BEA, Oracle, SAP, IONA, and others.

APPENDIX

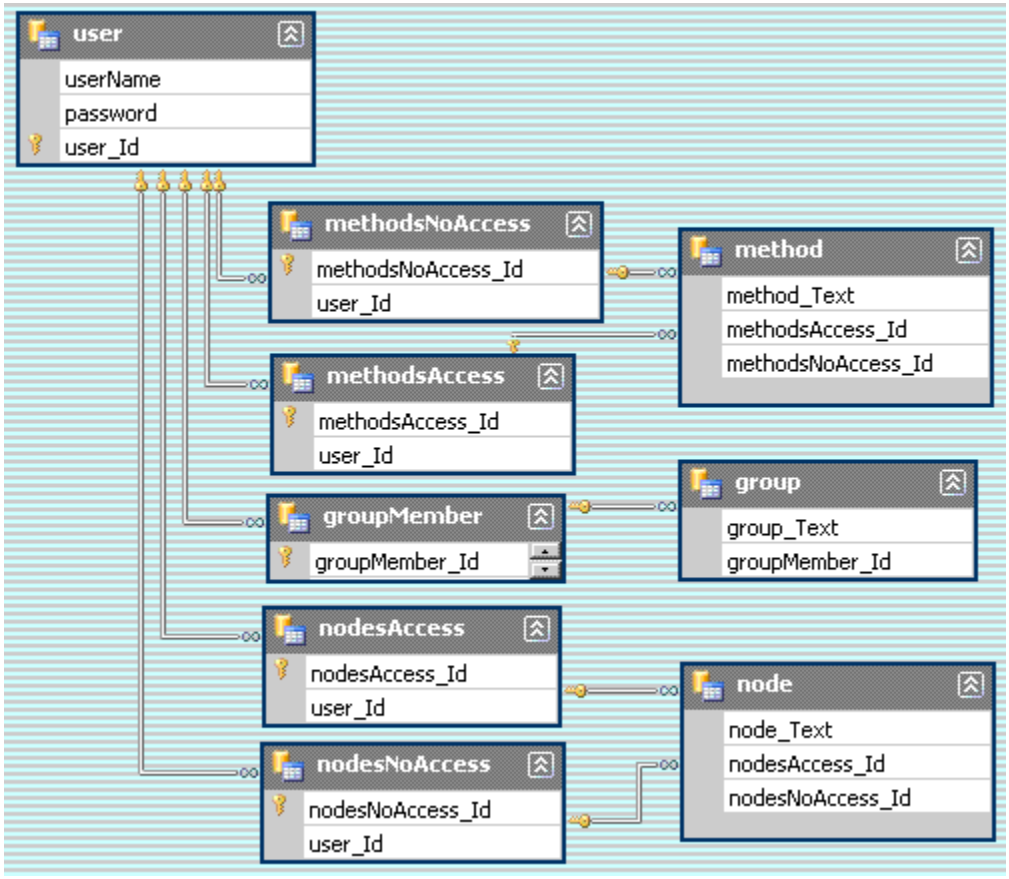
H AC

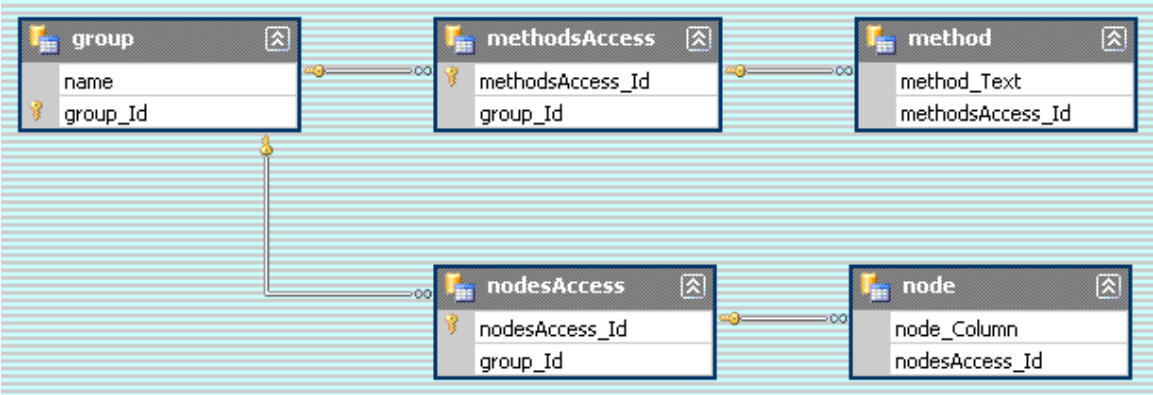
The AC is built on top of the path definition. Access can be granted or denied as either absolute path, or as a wildcard '*'

Read and write information is not set in the AC, but inside the function constraint settings.

Nodename eksample	Description
wpp	Absolute access to "wpp"
Wpp/node	Absolute node "Wpp/node"
Wpp/node.subnode	Absolute node "Wpp/node.subnode"
*	All wpp and nodes
Wpp/*	All nodes under wpp
Wpp/node.*	All node under wpp/node

Method must be granted explicitly.





Unit test AC

In creating unit test the AC was tested before integrating it into the big system. The test is depending on the UserInformation.XML and GroupAccessRules.XML files located in the bin dir of the test.
The unit test tests the following things:

- Login wrong username
- Login wrong password
- Logon ok username and password
- Check login for user GUID for login
- Check login for user GUID for not logged in
- Check login for random GUID

If confliction rules user rights has precedence

test	Group rights	User rights	comment
Method access	known	ok	Access
Method access	Unknown	ok	Access
Method access	Ok	Deny	noAccess
Method access	Unknown	unknown	noAccess
Method access	Unknown	unknown	noAccess
Node access	Ok	deny	noAccess
Node access root	Ok	Unknown	Access
Node access	Unknown	Ok	Access
Node access path	Ok	Unknown	Access
Node Access Wildcard	Unknown	Wildcard access	Access
Node Access Wildcard	Unknown	Wildcard access	Access
Node Access Wildcard	Wildcard access	Unknown	Access
Node Access Wildcard	Ok	Wildcard deny	noAccess

Node Access Wildcard	Unknown	unknown	noAccess

APPENDIX

I RISØ WPP

The data from the WPP at Risø comes as a binary stream over 3 serial connections. The data is placed in a text file contain a comma separated list. First a timestamp followed by 3 x 24 integers values as strings. The first 24 comes from com 3 the next com4 and the last com5. The stream is synchronized for every run through. The order of the numbers is important, and the calculation for then actual value depends on the positions of the number.

Channel	dau	col	c_type	type	name	Gain	offset	mapping (?? = unknown mapping)
Com 3								
28	1	1	.analog	Pconverter	POA	200	0	WGEN.W
25	2	2	.analog	SG	MzTT	1136,1	840	mz??
26	3	3	.analog	SG;EW	MxTB	4982,5	-1549	mx??
27	4	4	.analog	SG	MyTB	5114,9	-1234	my??
2	6	6	.analog	vane	WDRcos	0,222	0	Use in dau 7
3	7	7	.analog	vane	WDRSin	0,222	0	WNAC.WdSpd
5	8	8	.analog	termometer	T03	25	-75	WMET.MetAlt1.Tmp
6	9	9	.analog	baromenter	P03	92	600	WMET.MetAlt1.Pres
1	17	17	.digital	cup	WS36	0,624	0,177	WMET.HorWdSpd
29	18	18	.digital	status	Tip	1	0	Tip??
30	19	19	.digital	status	brake	1	0	brake??
31	20	20	.digital	status	gen	1	0	WGEN.GnOpMod
32	21	21	.digital	status	stall	1	0	stall??
7	22	22	.digital	rain(1/0)	Rain	1	0	rain??
Com 4								

121	1	25	.analog	SG	Mx12	-143,63	-32	Mx12??
122	2	26	.analog	SG	My12	123,6	-17,2	My12??
123	3	27	.analog	SG	Mx22	332	3,7	Mx22??
124	4	28	.analog	SG	My22	115,8	-235,6	My22??
125	5	29	.analog	SG	Mx32	150	128	Mx32??
126	6	30	.analog	SG	My32	117,4	-177	My32??
20	7	32	.analog	SG;torque	MxNr	193,2	80,4	MxNr??
21	8	33	.analog	SG	MzNr	55,833	6,198	MzNr??
22	9	31	.analog	SG	MyTB	51,539	48,652	MyTB??
Com 5								
8	1	49	.analog	Nacellepos	NP	72	103	WNAC.Dir
41	2	50	.analog	Refvolt	vane_ref	1	0	Used in Dau 3
42	3	51	.analog	SignalVolt	vane_sig	1	0	WNAC.WdDir
12	17	65	.digital	Cup	WSN	0,627	0,319	WMET.MetAlt1.HorWdSpd
11	21	69	.digital	Rotspeedfast	ssh	1	0	WROT.RotSpd
10	22	70	.digital	Rotspeedslow	ssl	1	0	WGEN.Spd

Formulas

If the value is analog it must be converted to a voltage -5V and 5V

Voltage $= \frac{10}{65536} * inp - 5$

Com	DAU	FORMULA
Com5	1 V1	$V1 * 72 + 103; 0 - 360°$
Com5	2 V2	$180 - (\frac{v3}{v2})357 + 1{,}5$
Com5	3 V3	

Com5	17 V17	$\frac{32000}{V17} * 0{,}62676 + 0{,}319$
Com5	21 inp21	$\frac{16000}{inp\ 21} * 60$
Com5	22 inp22	$\frac{16000}{inp\ 22} * 60$
Com3	6 V6	$\frac{180}{\pi} Arc\ \tan(V6, V7) + 25; 0 - 360°$
Com3	7 V7	
Com3	17 v17	$\frac{32000}{V17} * 0{,}62359 + 0{,}1769$
Everything else		*inp * gain + offest*

The numbers comes as DA values, and is calculated as described above. The conversion to the standard is done by finding a node in the standard that matches the provided node. It is not possible to find nodes for all the data. Unknown nodes are removed by the device communicator.

APPENDIX

J USEFUL PROGRAMS AND EXAMPLES

The .NET Sdk contains a number of useful tools.

svcConfigeditor used for setting up service

svctraceview for browsing service log files

svcutil. Generates stubs

WSDL generate stubs

xsd generate data contract

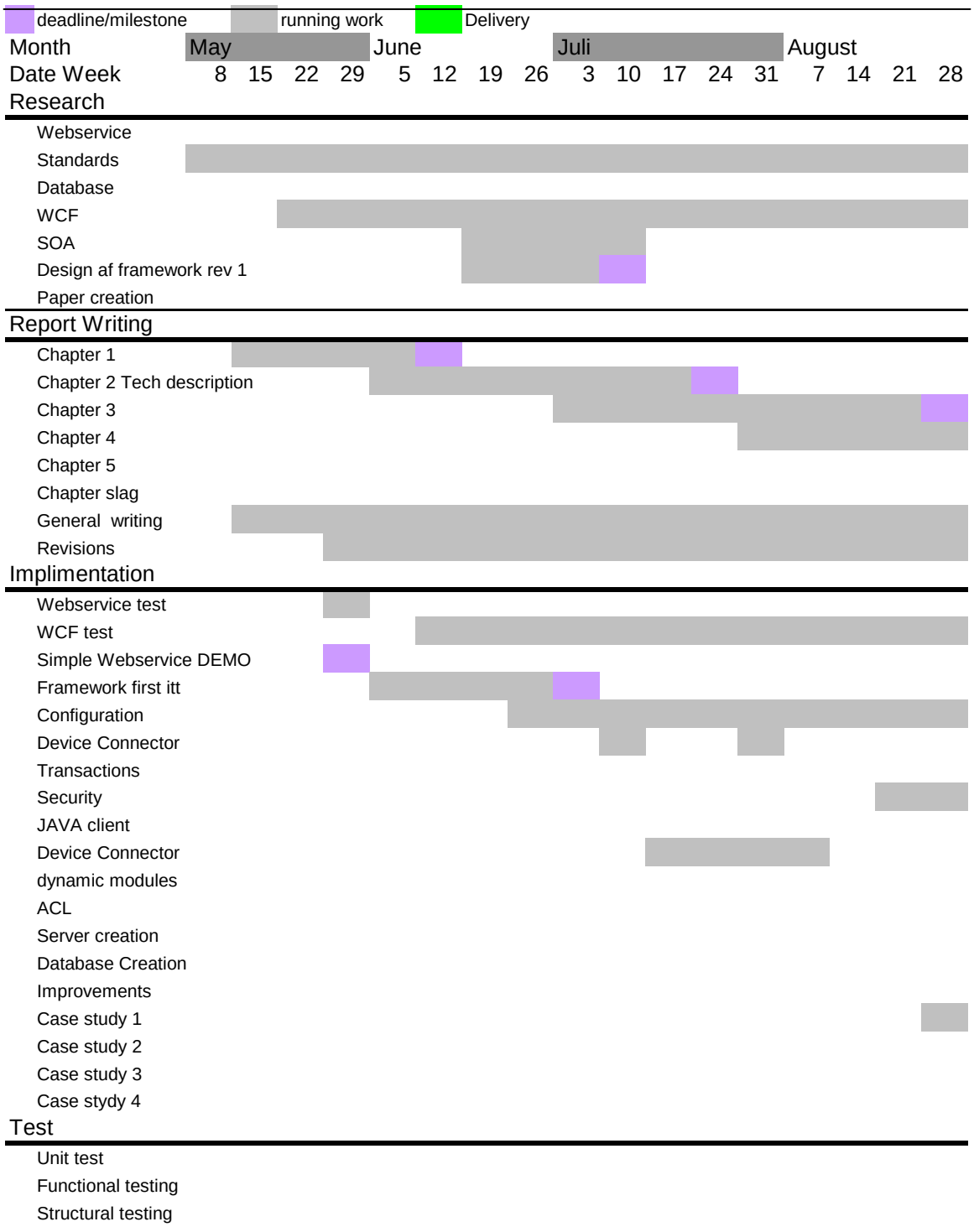
Some of the program snippets are self hosting, and will run out of the box, while others must be hosted on a server. The main inspiration and code provider has been the Microsoft SDK test case software. Unfortunately none of them worked without adjustments.

APPENDIX

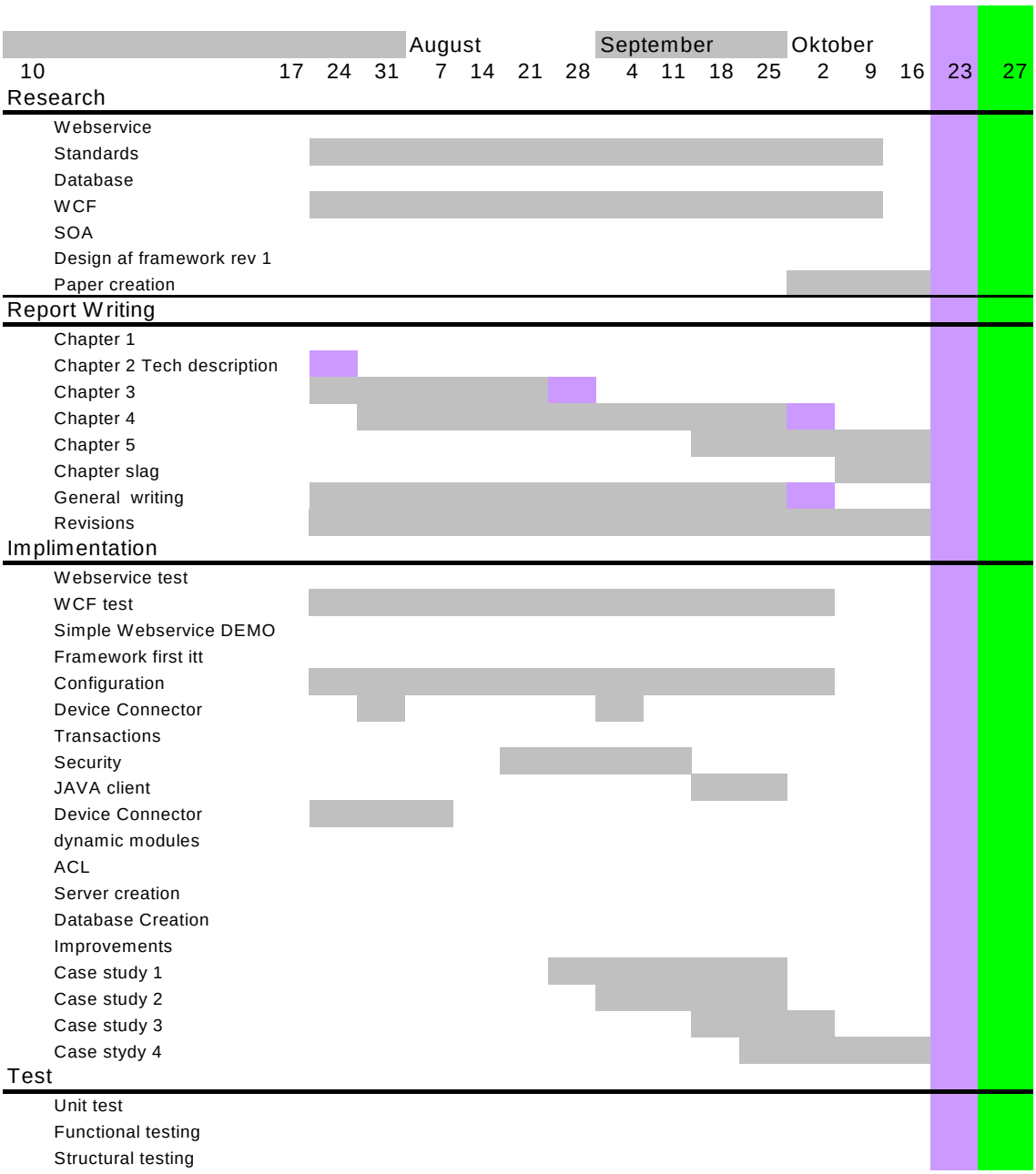
K TEST RESULTS

call http	response r	bindnding r	response o	calls	byte/call	reduction	calls more	choice http
text compr	908	2101306	1316821	2314	569,07	37,33%		
MTOM con	1512	3301734	1811503	2186	828,68	45,13%		
binary com	564	1435856	1308696	2545	514,22	8,86%		
text no con	908	2692414		2965			21,96%	
MTOM no	1512	4060758		2679			18,40%	
Binary no c	564	1692476		3001			15,19%	
text compr	767	1893803	1370313	2469	555,01	27,64%		
MTOM con	1255	2708814	1665401	2161	770,66	38,52%		
binary com	429	1103864	1174455	2573	456,45	-6,39%		
text no con	767	2433771		3173			22,19%	
MTOM no	1255	3599864		2871			24,73%	
Binary no c	429	1354400		3157			18,50%	
text compr	9514	6623003	852601	655	1301,68	87,13%		
MTOM con	9999	6220386	973539	623	1562,66	84,35%		
binary com	5955	4024412	806842	673	1198,87	79,95%		
text no con	9514	6765301		712			8,01%	
MTOM no	9999	6710337		672			7,29%	
binary com	5988	4341776		726			7,30%	

call tcp	response r	bindnding r	response o	calls	byte/call	reduction	calls more	choice http
text compr	1104	3198111	1838161	2897	634,51	42,52%		20,12%
MTOM con	1592	3987646	2144445	2507	855,38	46,22%		12,80%
binary com	571	1675226	1549330	2934	528,06	7,52%		13,26%
text no con	1104	4116639		3729			22,31%	20,49%
MTOM no	1592	5154582		3240			22,62%	17,31%
binary no c	571	2146301		3759			21,95%	20,16%
text compr	1006,5	3117229	1817273	3097	586,78	41,70%		20,28%
MTOM con	1335	3655994	2192603	2741	799,93	40,03%		21,16%
binary com	436	1392195	1502164	3193	470,46	-7,90%		19,42%
text no con	1006,5	3978793		3953			21,65%	19,73%
MTOM no	1335	4683944		3511			21,93%	18,23%
binary no c	436	1802035		4133			22,74%	23,61%
text compr	9594	6448095	892071	673	1325,51	86,17%		2,67%
MTOM con	10079	6502043	1024998	646	1586,68	84,24%		3,56%
binary com	5995	4202978	851515	702	1212,98	79,74%		4,13%
text no con	9594	7311555		763			11,80%	6,68%
MTOM no	10079	7247889		720			10,28%	6,67%
binary no c	5995	4604643		769			8,71%	5,59%



Time table continued


August
September
Oktober
10
17
24
31
7
14
21
28
4
11
18
25
2
9
16
23
27
Research
Webservice
Standards
Database
WCF
SOA
Design af framework rev 1
Paper creation
Report Writing
Chapter 1
Chapter 2 Tech description
Chapter 3
Chapter 4
Chapter 5
Chapter slag
General writing
Revisions
Implimentation
Webservice test
WCF test
Simple Webservice DEMO
Framework first itt
Configuration
Device Connector
Transactions
Security
JAVA client
Device Connector
dynamic modules
ACL
Server creation
Database Creation
Improvements
Case study 1
Case study 2
Case study 3
Case stydy 4
Test
Unit test
Functional testing
Structural testing

APPENDIX

L PROCEDINGS PAPER

PROTOTYPE OF GENERIC SERVER FOR WIND POWER PLANTS USING IEC 61400-25 STANDARD

Andreas Kargård Olsen, Baris Ösdil, Bjarne Poulsen, Knud Ole Helgesen Pedersen, Informatics and Mathematical Modelling, Centre for Electric Technology .
Technical University of Denmark Building 322, office 007, DK- Kongens Lyngby:Denmark
Phone (45)4525 5274, Fax +4545255300
E-mail:bjp@imm.dtu.dk & khp@oersted.dtu.dk
Knud Johansen. Q-Technology

Keywords: IEC 61400-25 Standard, Communication, Generic Server, Web Service, Wind Power Plant, Data Model, Service Oriented Architecture, Framework

ABSTRACT

IEC61850[1] has defined a family of standards for the power grid. For instance, the new IEC 61400-25[2] defines protocols for communication, control, and monitoring of wind power plants (WPP). This standard includes a wide range of mandatory and optional elements in the defined models, ranging from interfaces for control and monitoring to a standardized and secure way of handling communication. An analysis focusing on isolating the necessary requirements has been carried out based on the IEC61400-25 in order to create a generic prototype which can be used by WPP vendors. The main communications interface of the prototype utilizes web services and the prototype developed is comprised of several independent modules to allow for the possibility of choosing a fully customizable setup by the end user. Configuration of the system needs to be done in a simple way, ensuring a flexible and reusable system, where different choices for the system can be added or left out depending on user specifications. From the requirements a prototype with the purpose of examining the key aspects of these definitions has been elaborated.

1. INTRODUCTION

Wind power plants have through the years steadily gained a bigger and more dominant position in the power generation industry. Each vendor has their proprietary solutions on controlling and monitoring of the products supplied. In today's ever changing and rapidly growing energy market, monitoring of and easy communication between different systems, is essential. Through this communication the current state of the individual power plant can be controlled and monitored when required, and counter measurements can be enforced if needed in order to meet the changing demand for energy and keep the stability of the distribution system. It is vital that the overall dispatching systems are able to control the energy generation from a wind farm on demand in order to meet the fluctuations in the energy consumption. A common way to achieve this is a vendor independent approach.

As the complexity of the power distribution network increases, methods for efficient analysis, monitoring and coordination of the network control become essential. This in turn requires a highly efficient and dynamic control strategy for the power system network. Several organizations have addressed this issue in manners where the main objective has been to develop communication standards for interconnecting electric power generation systems.

One important effort towards standardization has been launched by the International Electrotechnical Commission, IEC. From the start IEC focuses on communication within substations has led to the IEC 61850 communication standard. Several standards for distributed power generation are under development using the structure of the IEC 61850 standard.

The vision of this project is to evaluate the new standard 61400-25 for the WPPs and end up with a functional prototype implementation of the standard

A framework encapsulating and compliant to the standard must be designed. This framework must expose the methods offered in the standard and taking care of all communication. The framework should be transparent so that a programmer can change the configuration of the framework. The approach for adding different layers to communication must be done on top of the communication model to ensure that different approaches can be used, for instance to secure communication, the framework will do most of the work involved in the transition automatically.

The power distribution system is comprised of many different devices, distributed over a great physical space and requiring a variety of functionalities. SOA is a system architecture where a network is comprised of nodes exposing services to each other in order to complete a greater common task. The system will take advantage of this architecture where appropriate. Every WPP will be modeled as a web service taking care of the communication with different clients.

For the time being there has not been specified a common way to configure the system base on 61400-25 standard or its constituting devices. In the future an extension to basic parts of standard will be released describing a configuration lan-

guage. The prototype must be designed in a way where everything can be configured through a common interface.

The aspect of security is a major part of a system communicating data over the network. Security must be a configurationally option.

A massive amount of data is collected form the WPPs. These data has to be analyzed, filtered, and made accessible to clients as fast as possible. This leaves a lot of constraints on the storage device pertaining to access time, storage size and stability among others. In reflection of this, a great deal of work must be put into the design of fast access to the data and the storage structure making sure that it can meet the specifications that a system base on 61400-25 standard demands.

2. ANALYSIS OF THE STANDARD AND THE REQUIREMENTS

The IEC 61400-25 series is a specialized version for defining and standardizing a unified communication for monitoring and control of wind power plants. One aim is to enable systems from different vendors to mutually communicate.

The IEC 61400-25 series is an extension of the previous IEC 61850 series of standards, which in general defines communication networks and systems in substations. IEC 61400-25 does not simply replicate IEC 61850 but reuses the definitions which in general apply to all power systems

The standardization expands over the information modelling of the target system and the communication protocol for communicating the data encapsulated in the information model. As a result of this approach, the standard addresses the domain by separating it into three main categories of interest which together encompass all the important aspects of the communication and control of wind power plants. The three different main categories are as follows:

1. Wind power plants information models.(IEC 61400-25-2)
2. Information exchange models. (IEC 61400-25-3)
3. Mapping to communication profile. (IEC 61400-25-4)

2.1. Information Models – IEC 61400-25-2

The information model uses an Object Oriented approach for modelling the wind power plant components and data in general. It defines an exact model of the wind power plant (WPP) which contains the components and data of interest. This data will be made available to access for monitoring and control purposes. The information model constitutes a precisely defined set of reusable data classes which will make it possible to build up a logical model of a specific WPP. The primary component in the model is a Logical Device (LD) which is defined as an abstract model able to fully represent a WPP. A LD residing on a server is assigned to a specific WPP which must be able to fully represent it with all its data and attributes. It must

contain a collection of specific Logical Nodes (LN) which will further contain data instances reflecting the physical state of the WPP. Logical Device is a container for all WPP related data together with self descriptive meta-data describing the physical host and the device itself.

A logical node consists of a collection of related data, defined as Data Classes (DC). All the information in a logical node is contained in respective Data Classes. The structure of all logical nodes is similar and has a standardized form where different types of logical nodes can be constructed through the combination of different optional data classes.

All the logical nodes used in modelling the WPP inherit their structure from the abstract logical node class defined in IEC 61850-7-2. From an implementation point of view these different logical nodes will be similar since the structure is based on a common definition and follows a common pattern.

The hierarchical structure of a logical device residing on a server together with logical nodes and data classes is depicted in the figure 1 below.

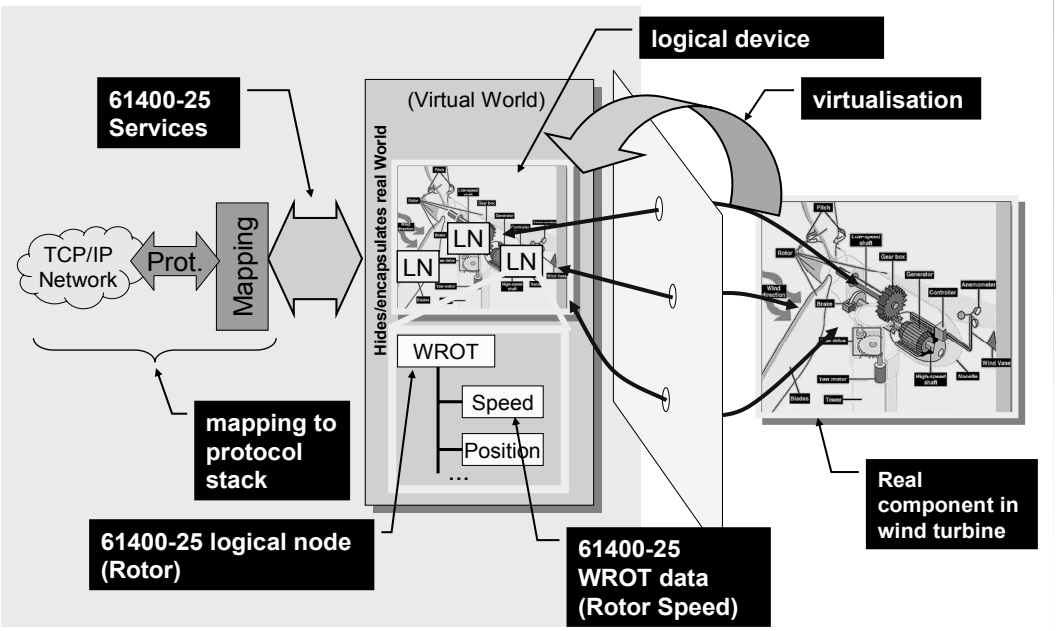


Figure 1. Modeling approach (IEC 61400-25-1)

The data class is the actual component of the information model which is used to define any data contained in logical nodes. These specific classes are called common data classes, which are precisely defined classes inherited from data classes.

The common data classes used to model a wind power plant device can mainly be categorized under two groups. Common data classes a), defined specifically for wind power plants and b), logical nodes that are inherited from IEC 61850-7-3. A complete listing of these common data classes is provided in IEC 61400-25-2 clause 7.

The server must be capable of representing the information model with all its instances from logical devices and all the way down to specific data attributes. The hierarchical structure must be preserved so that each data instance can be referenced in a standard manner as defined in the information model.

A server may host one or more logical devices depending on the number of WPP which must be controlled by that server. Therefore it must be possible to uniquely reference a specific logical device representing a specific WPP.

The server must be capable of representing a logical node contained in a specific logical device and to refer to it uniquely. Each logical node has a unique reference name which makes it possible to locate it directly in conjunction with unique reference name of the logical device containing the logical node. This is important since a single server instance can have more than one logical device residing on it.

2.2. Information Exchange Models IEC 61400-25-3

This part of the standard describes the information exchange model which is implemented on the server enabling client's systems to access and modify data in the information model. Each information model instance has a service interface describing the operations available on that particular instance. Each information model object has a specific set of services making it possible to read or write from/to it.

The basic services that are used to mediate between the outside world and the real wind power plant device are referred to as Abstract Communication Service Interface (ACSI). The basics of these services described in details in IEC 61850-7-1 and IEC 61850-7-2

In the ACSI models, the information that gets reported or logged is represented by data sets. In that way reporting and logging can be defined in a more compact manner applying to a group of data.

The reporting services must make it possible to subscribe to spontaneous data reports on specific conditions for data values. Conditions such as change of value or change of attribute values will trigger a preconfigured reporting subscription and start dispatching the values.

In the reporting mechanism, the server must make the data for the client available for read and write. Since it is not common practice in client server architecture for a server to contact a client offering data, the server should buffer the values to deliver them later to the client, whenever a client request is made.

In order to achieve buffering mechanism so that the server does not have to notify the client for the available new data, some sort of a server side session must be implemented for the reporting interaction between client and server. The server must have an internal state making it possible to keep track of which step in the reporting process it is in, and which data has been sent and which is still in the buffer. However, the standard also specifies that it must be possible to configure the reporting mechanism such that data is not buffered in case of a connection interruption, meaning that the client only can access the data available at the time it makes its service request.

Logging like reports can be initiated upon the client's request. Like reports, the data can be logged on the same criteria, but in addition to this, updates also must be logged.

The client must at any time be able to get a log stream for a given interval. Due to the data storage amount it must be assumed that this interval is limited. Reports reflect current data while logging reflect longer-term data, and system status.

2.3. Mapping to communication profile IEC 61400-25-4

The services defined in the information exchange model are mapped to standard web services. A detailed description of each service is provided together with the corresponding WSDL [3][2] describing the exact structure of the service methods. Each service defined for the various data models is mapped to SOAP services making it possible transfer data with the correct types and structure defined in the information exchange model.

The server must resemble the web service description provided in the communication profile mapping. It should be possible to communicate between client and server accordingly as specified in the information exchange model.

3. CHALLENGES AND DESIGN REQUIREMENTS

The standards do not provide a solution for every single aspect of implementation and in many cases the standard leaves several choices or decisions for the system designer to solve. A presentation of the challenges in designing and implementing will be given next, together with a suitable solution strategy addressing the identified problems for the design of system based on 61400-25 [4].

3.1. A framework based on components

The framework must take care of all the interaction with the protocol. Security, reliability, sessions et cetera. must be implemented in a standardized way. The framework must have a default setting for communication, but it should still be possible to change the different settings as needed. Looking at end-to-end communication, there are many single elements that must be easily changeable. For instance, each WPP might have its own way to supply the data. The system must be a

very-late binding architecture, where one component's runtime is integrated with another component's runtime using dynamic invocation.

The system is to be designed using multiple modules that can be changed on demand. This implies that a common interface must be used. No matter how the mapping in the transport is done, the resulting requests and responses must be the same.

3.2. Configuration

The information model defined in IEC61400-25-2 has a lot of optional nodes that can be implemented with optional data classes. The vendor must have a way to configure what their WPP has implemented, and how the communication is done. Providing such a configuration option will make reuse more efficient and it will be easy to set up a server representing specific WPPs. Configuration also includes information for how interaction between the WPP and the system must be performed, for instance how often can/must the data be pooled from the WPP.

The information model does not contain information of *how* the WPP and the system will communicate. It only states an abstract data format. Each vendor would have to design an entire system from information model to exchange model unless a common data input format is defined.

3.3. Data processing and data storage

To make a system that can handle the massive amount of data flow it is vital that data can be processed in an efficient and secure way. The data must also be rapidly retrievable and easy to store.

On the server side, the information model must accurately reflect the hierarchical information structure. It is extremely difficult to represent the model using a classic relational database. Besides the difficulty in representing the model, data retrieval will also be extremely slow. This is because the server must execute a large number of queries to retrieve data from the complex relations in the data hierarchy. Instead of storing the current data on a persistent storage for service retrievals, the information model could be constructed on the server process itself making use of the representational power of an object-oriented language. The server process could programmatically replicate the information model and store the data in its run-time with suitable objects while running and servicing the client systems. This approach will make the execution very fast since the data is already stored in main memory ready to be fetched. And the modelling/representation of the information will be straightforward through instantiating objects from class definitions resembling the class definitions in the information model.

To store the most current data and make it available for client processes, the server must continuously pool the WPP for new data made available. Only the latest data is kept inside the system to speed up data handling. Long-term data is separated from the running system to cut down runtime challenges. The information model is

modelled in an object-oriented way, to keep the implementation and the information model close together.

3.4. Security

IEC 61400-25-3 defines the security aspects for the standard and how to solve it in general, but how it is handled specifically is completely up to the individual supplier.

One supplier might simply use a secure line, and therefore remove any security aspects of the service itself, while another might want to use a public ISP where the service related traffic must be secured by the service itself. This calls for a solution where security is built on top of the communication as a separate layer in a modular fashion easy to add, remove or change on demand

In TC 57 [5] a proposal has been presented in which a security model has been suggested.

In this research TCP/IP traffic is utilized, and therefore has to follow the IEC 62351-3 security standard. The standard recommends transport layer security (TLS) to tackle the most common security threats. At the same time it specifies that security must follow progress and update to better solutions when available.

Another aspect of security is access control. Access control has the duty of ensuring that only authorized individuals can gain access to the data. How and what the security includes is defined by the service mappings (SCSM).

In IEC 61400-25 the minimum requirement for access control is only defined as the need for supplying a valid username and password to gain access. This ensures that only people with a valid password can gain access to the system. Different users might have the rights to perform different actions.

The system only contains methods and data the client can access. The user either has the right to run a method or not. Through the method the client can either read or write data by functional constraints. Every client does not necessarily have the right to view all data but maybe only a subset.

Users can be granted read or write rights to specific nodes. In the same way users can get clearance to invoke specific methods. Each node or method must have lists of what each user can and cannot do.

4. PROPOSALS FOR A GENERIC ARCHITECTURE

To make the system generic, different elements in the system must be isolated. The figure below shows a proposal for the system architecture.

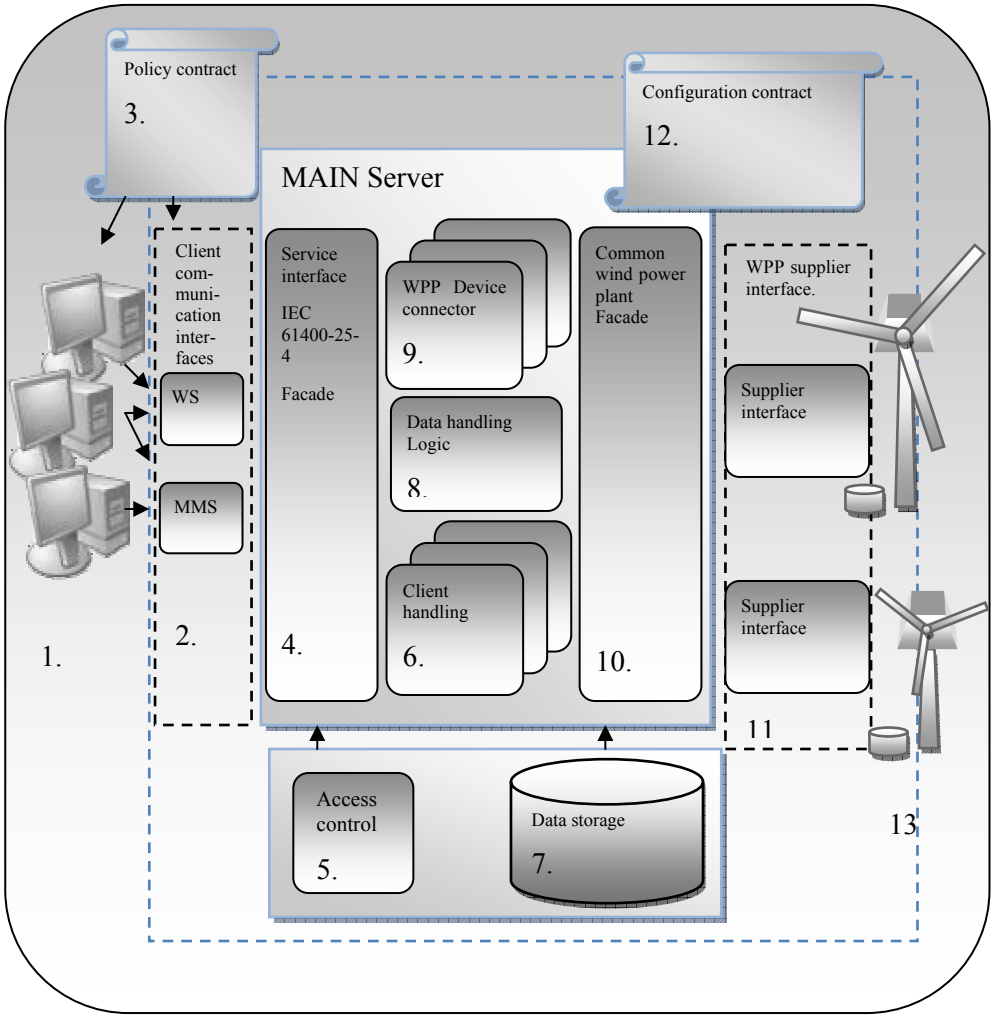


Figure 2. Proposal for a system architecture.

1. Clients can be web clients, applications et cetera. (The client is outside the scope, and should simply be an end user of the service.)

2. The client can use different methods to conduct the communication with the service. The main focus is on web services but it can equally be an MMS or others. The interface is defined in IEC 61400-25-4

3. Policy files define specific information for the communication. This includes security information, use of transport protocol, et cetera. Policy files are ideal for custom configuration at Deploy time.

4. A common interface assures that the service can always be reached through the client communication interface as long as the client obligates himself to use the interface defined in IEC61400-25-4.

5. The access control checks the client's right to access a specific method or data-set.

6. When communicating with the service the client can ask for specific data. The client handler manages the connection between different clients and the service. The information exchange model defines how this is done in IEC61400-25-3

7. Stores the data for shorter or longer time.

8. Data handler logic keeps track of data, and the logic to run the service generally.

9. Different WPP suppliers produce different data. The Pooler gathers the necessary data from each WPP. The information model in IEC 61400-2 defines this.

10. There is no generally accepted standard for how data must be supplied by the WPP, but by defining one, supplier modules are easier to replace in the system.

11. Supplier interface. When connecting a WPP to the service, the supplier has to create an interface file. The file has knowledge of the supplied nodes that the WPP can expose, and how often pooling is necessary.

12. The contract must contain information of the data each WPP can deliver. Also the contract defines how often the data is to be pooled. This must be done in compliance to the IEC 61400-25-2 information model.

13. Each supplier exposes its capabilities to get and set data through their own interface.

5. CASE STUDIES

The purpose of these case studies is to try some of the different scenarios that the system is supposed to work under. The overall design of the system has been built in order to have the same basic design for all the challenges, and then make minor changes in order to add or remove functionalities to the system. Before studying each case, the setup, overall architecture and the internal workings of the server system are described. The prototype focuses on web services.

5.1. System Description

The core unit of the system is a server process hidden behind the service interface where device data and client requests are handled. It has two important interfaces through which it accomplishes its tasks. These are, respectively, first the interface to the communication service which acts as a bridge between client processes and the server, and second, an interface which can be configured to pool data from WWP. The data retrieved from the configured device(s) are kept in an internal, in-memory data model for fast retrieval and processing when client requests are being handled.

IEC 61400-25 defines an event driven service model for monitoring WWP. Especially in the reporting process various conditions have to be monitored continuously as new data arrives, so that proper actions can be taken in order to satisfy each report configuration which will serve the client process with information.

Service Interface - Communication Module

The Service is supposed to work with different client protocols. One of these is Web services, but the system should also be able to work with other protocols like MMS. It is possible to implement the protocol directly in the server, but a more flexible solution is to have one server to serve the different communication modules through a well known protocol. A facade presenting an interface can be created. Each protocol then has to implement a module that takes care of getting information from the client and returning the server reply to the client. This conforms to the data defined in IEC61400-25-3. This approach enables a more modular architecture for the system

Device Interface

The communication interface to WWP from the main server is beyond the scope of the IEC61400-25 series. No specification is given on how to handle the communication with devices that are supposed to be monitored and controlled. However, in order to be able to test the server prototype, a simulator is implemented and connected to the system. The simulator acts as a device continuously generating data for a specified set of data objects, for a pre-specified logical device data model defined on the server. The data is extracted from log files which contains real-time stamped wind power plant device data. The data entries contained in the log files are mapped to data types defined in IEC61400-25 where suitable.

Server Configuration

Initially at start up, the server must be configured in order to work properly and interact with devices and communication services attached to it. At start up it will configure itself through an XML configuration file. The information model reflecting device data, device connections, pre-configured data sets and access control configurations are all made through dedicated XML files for each specific configuration.

Configuring the information model

At start up the server will first create the data structures reflecting each specific device attached to it in order to be able to house the data and make it available to client systems. The XML configuration file makes it possible for a server administrator to create data structures for logical devices with its logical nodes, data objects and data attributes. Initially the configuration file defining the data structure with their default values is parsed, where afterwards the object instances are created on the server according to the structure defined in the configuration file. It is possible to initiate data structures for several logical devices in the same configuration file.

The configuration file resides with the server. The server must know the exact data structure of the physical devices which are to be attached to the server before composing the configuration file. In the future the configuration could be done dynamically by retrieving configuration information directly from the physical device itself.

5.2. Case study 1

This case study has been focused on the following subjects:

- Development of a generic device communicator for acquiring data from different WPP data suppliers
- Securing access and data across a web service
- Cross-language access, and different clients

This case has shown that web service clients in both Java and WCF can be connected to the web service, and retrieve the information gathered for different WPPs. It is also shown that a web client can gain access to the service though the use of a web server.

The device connectors could act by it self or represent several WPPs. The gathered data could be collected and managed in one central server.

Security can be addressed in a seamless and uncomplicated fashion following security patterns.

5.3. Case study 2

In this case study, the focus is on a virtual major corporation owning a big wind farm. They operate many WPPs and are monitoring the system from one or more locations. Some of the WPPs are connected using a modem. For this reason it is important to get the most out of the bandwidth by compressing the data transmitted. In this scenario, the communication line between the WPPs and the server can be lost. In case of connection loss it must be possible to recover the data produced during the outage.

Some of the WPPs have their own device connectors and others have a joint connector. Within the corporation many people must be able to view the data simultaneously. The corporation has a dedicated secure private line for internal communication. No security is needed concerning this of the traffic. However, data must be protected internally. Not everyone in the corporation is allowed to access data, and even if they are allowed, not everyone is allowed to run all service methods on the WPP devices. Access control via a replicable module is included.

Key points:

- Many WPPs are hosted under one server.
- Multiple clients.
- Avoiding data loss during offline period.

- Compression for lower bandwidth usage.
- Role based access control.

Offline scenario

In the case of loss of connection, data must be stored until the client can retrieve it. An extra or redundant server connection could provide an alternative way to upload data, but since the WPP is supposed to be autonomous and recover from critical situation itself, it must be concluded that information is not time critical at a level , and this solution is routed out, in order to not have the extra cost of maintaining several communication lines. A much cheaper method is to use the hardware already placed at the WPP. In such a case each device connector keeps a buffer of data which can be retrieved on demand. However, if the data is not retrieved by the client, the list will keep on growing in memory and eventually consume all of the memory resources.

To overcome this problem data must be swapped to disk on regular intervals. This will not only reduce the memory usage, but it will also assure that data is not lost if the machine hosting the data connector crashes.

The device connector works with time stamped packages. If the buffer holds more than a given number of entries the data will be saved to disk, and the stack will be emptied. On the next attempt by the server to retrieve data, the retrieved object will contain a flag telling the server that buffered data exists. The server can retrieve this data on demand. Both size of the stack, file size and transferral web service transferral buffer must be configurable.

If the connection is poor, but do exist, reliable sessions are a great addition to the quality of the communication. “Reliable sessions” or “reliable messaging” assures that a message will be re-sent if communication is disrupted. It will keep on trying until the message has been delivered, but at the same time uphold an assurance of exactly-once and in-order delivery of the messages. Reliable messages are part of several of the standard bindings, and can be turned on and off on by demand. Like other attributes, this can be done either in code or in the configuration, by enabling reliable sessions. Reliable messaging is turned on for the prototype.

Access control (AC)

To ensure that only the right people have access to data, an access control system is needed. The standard proposes an access control system, but it has not yet been fully specified. For the purpose of the prototype, a system for controlling the access to the system has been implemented. This is built as a role base access control system. (RBAC)

Ensuring module exchangeability

The modules in the system should be interchangeable Modules are accessed from many different methods in the server, and as such they are incorporated deeply in the system. The solution lies in linking the object during start-up rather than under compilation. The principal is described by Martin Fowler as a plug-in pattern [6]. This pattern is one of the corner stones in order to have interchangeable dynamic modules. A negative effect is that DLL access is slower than if the linking was done at compile time. But this is a minor expense compared to the big flexibility it provides for the system. The downside to this approach is that all modules must conform to the same module interface, and therefore the interface must be finalized.

Minimizing traffic

SOAP is not a good format when considering the amount of traffic generated. It is in plain-text format and contains lots of meta-data to describe the message. This makes SOAP a perfect candidate for compression.

For web services using WCF there are three main ways to encode the data for transport. These are text, binary and MTOM [7]. Text is plain SOAP in ASCII encoding; this of course is the most compatible way to send the data, but also the most expensive with respect to bandwidth consumption. MTOM is a WS standard, and can compress the body of the message. MTOM can also be used for transporting binary data between services. This comes in handy if files should be transported from a server to a client, but since it still has to be transformed to base64 encoded data, and still uses SOAP wrapping, alternative methods like FTP transport would probably be a better solution for this. Different bindings or compression do reduce the traffic drastically, but it does slow down the system because of the extra computation needed [8][4]

Binary traffic is the most optimal way to transport the data, but is restricted to a windows-to-windows communication. However if the communication is restricted to windows-to-windows communication, this is the best choice

The last approach for reducing traffic is to create a custom encoding schema. For instance, the traffic could be encoded using one of the commonly used compression libraries, like ZIP or RAR. This of course will demand that both the client and the service are able to encode and decode the traffic. Compression is specified in the HTTP 1.1 protocol.

Simulating the wind farm

Each device connector must be initialised at start-up. The server knows where and how and when data must be retrieved, and will in turn collect it.

The clients contact the service through the Internet information server. This server assures that multiple clients can interact with the service at the same time.

Case summary

A device connector module is able to store data for later retrieval from the server. The module provides a way to get data whenever possible, in manageable data packages. This ensures data delivery, and at the same time reduces peek loads on both the device connector module and the server. No standard is defined for how the data is conveyed from the physical device to the server, but the connector (device simulator) did provide some basic cases that all connector should address no matter how they are designed internally. The lack of a standard for a WPP communication makes it hard to reuse the system between different vendors, because the module will be incorporated into the server, unless a common interface is defined. However, with a defined interface at hand, the system would be very flexible and have opportunities for changing the modules as needed.

5.4. Case study 3

Case study 3 looks at some of the different bindings and protocols that can be used in the communication.

The main aspects of the case study are:

- Tests different encoding with different security choices and compression.
- Relative speed
- Relative number of calls serviced
- Multiple end points for better flexibility and performance for the clients.

It is worth remembering that most of the bindings support the same basic features like security, reliable messages, different encoding schemes, and as such the basic features can be kept no matter which binding is chosen.

A test shows that the different standard encodings did influence both speed and data amount sent. The test did show that the system did react to different configuration parameters.

Due to the nature of web services and the way they are configured by using for instance Windows Communication Foundation (WCF) [9][10], it is easy and cheap to provide several ways for clients to communicate with the system.

Additional connection policies can be added simply by supplying an endpoint to the configuration file. Not all clients support all of the possible choices the service can communicate. By exposing multiple endpoints each client can choose that endpoint providing the highest efficiency for him. This ensures that the system does not have to operate only under conditions designed for a worst case scenario, but it can provide different levels of service. This maximises flexibility for both client and service providers.

5.5. Case study 4

Part 5 of the IEC61400-25 series is dedicated to conformance testing. The completed system should be tested successfully against all the proposed tests. What to test for varies a great deal but the most common attributes a system should have includes capability, reliability, efficiency, portability, maintainability, compatibility and usability.

The tests conducted for the developed prototype is limited to the functional testing of the services offered and general unit testing during development of the prototype. The aim of functional testing was to test whether the service methods operated as they should with proper request and reply objects.

The service tests conducted shows that the implemented service methods can generate the expected responses with the correct data types defined for the services. This implies that the business logic executed on the server is correct as well.

6. CONCLUSION

The goal was to implement and test an IEC 61400-25 compliant generic server which can be used to monitor and control wind power plants. During the development of the server there has been used real log data from wind power plants.

The main focus has been on the server system and its interaction with clients through the specified web services defined in IEC 61400-25-4. When developing the server system, general considerations such as security, modularized software architecture has been taken into account and implemented, whenever applicable.

Almost all services defined in the standard have been implemented using web services. This includes mapping and representation of data on the server side and to communicate the data to client systems.

The IEC 61400-25 standardization series is still “work in progress”, and during this research several updates have been released. The new releases of the standard were mainly dealt with ACSI service mappings to a communication profile. This is part-4 of the series defining the web service interface with its data types. At a certain stage the project had to be closed for changes at some stage in order to finish the implementation. However, services implemented lately are in conformance with the later versions of the mappings.

The list below outlines the main requirements which have been implemented in this project.

Information Model
On the main server, a suitable data structure has been set up which can store data as defined in IEC 61400-25-2. It is an in-memory data structure making data manipulation and retrieval very fast and efficient. When building logical devices from logical nodes and data instances, the exact hierarchical structure can be presented

on the server. The information model on the server is initialized through an XML file which makes it possible to alter the data models of the different devices easily.

Service interface
The web service interface, through which the service methods are communicated, has been implemented. The web service is implemented from the WSDL given in IEC 61400-25-4. WCF has been used for the web service in order to benefit from latest improvements within web services.

Different end points providing different communication features can be set up for the service so that client systems can choose the best suitable endpoint. Tests in case study 3 shows that different choices regarding communication (encoding, encryption, and transport protocol) will result in big performance variations.

Services
The server logic for the services has been implemented. Business logic within the different modules has been isolated from the server. It is possible to send service request with the correct data types defined in the standard and retrieve the corresponding response.

Module based architecture
The server components have been implemented as separate modules which make it easy to configure and change the system. For example the service interface is not tightly coupled with the central server itself.

It is possible to attach another service interface using another protocol than SOAP/web services.

The change of the AC module is only a matter of replacing a DLL in a directory. It is not even necessary to provide any configuration, changes, the system will automatically recognize that a module containing AC information is present, and it will automatically load it into the running system.

Reporting
The reporting service has been implemented according to the event driven data retrieval model defined in the standard. A client system can subscribe for reports which are generated by the report control blocks initialized by the subscription service. Both buffered and un-buffered reporting has been implemented.

Access Control
An access control mechanism is implemented making it possible to create a specific view of the data and services applied to it. The access control module can be configured so that different views of the system can be assigned to different client systems.

Device simulators/connectors
In order to be able to test the system, device simulators capable of generating data have been implemented. The simulators uses log files from real wind power plants

when generating data. Different scenarios, which are most likely to appear in a real world case, have been studied, such as connection loss and lost data packages.

Configuration

The server and its modules are implemented such that they can be configured easily. For example the access control mechanism can be configured through XML files in order to create different views of the system.

The server can be configured to handle different WPP simulators with different data models. The structure of each simulator attached to the system can be defined separately.

The device connectors can be configured so that the server can locate the simulators and associate it with the respective data model defined on the server.
The security and transport features of the system can easily be configured through configuration files.

Clients

A web based client has been developed to test and demonstrate the services implemented. The implemented communication module for the server is based on web service. Any client capable of consuming web services can connect to the server and perform its tasks through the web services made available.

Overall Conclusion

The main focus was on creating a system to implement a server system compliant with IEC 61400-25 standardization series, a system that was not tightly bound to specific technologies and easy to configure. The resulting implemented server system has shown that such a system is possible.

The system covers most of the major parts defined in the standard. All topics in the requirements has been addressed at some level, and we believe this research should provide a good basis for future work and efforts toward the great goal of creating a vendor independent communication environment for all wind power plants in the world.

6. ACKNOWLEDGEMENT

We gratefully acknowledges the support from Claus Bjerge E2 part of DONG energy for real data from wind farms and the contributing with knowledge about Wind Power Plants. We would also like to acknowledges Aksel Kargaard Olsen for reviewing and commenting on this document.

7. REFERENCES

[1] http://www.61850.com/

[2]http://www.iec.ch/cgi-bin/procgi.pl/www/iecwww.p?wwwlang=E&wwwprog=sea22.p&search=iecnumber&header=IEC&pubno=61400&part=&se=&submit=Submit

[3]Web service description language http://www.w3.org/TR/wsdl20/

[4]Andreas K.Olsen, Baris Özdil Prototype for a IEC 61400-25 Compliant Generic Server

[5] Frances Cleveland IEC TC57 Security Standards for the Power System's Information Infrastructure – Beyond Simple Encryption

[6] Martin fowler Patterns of Enterprise Application Architecture.2002

[7]SOAP Message Transmission Optimization Mechanism http://www.w3.org/TR/soap12-mtom/

[8]Christopher Kohlhoff Evaluationg SOAP for High Performance Business Applications: Real-Time Trading Systems.

[9] Keith Brown security Brifes http://msdn.microsoft.com/msdnmag/issues/06/08/SecurityBriefs/default.aspx#S3

[10]David Pollmann Programming INDIGO