TAMPERE UNIVERSITY OF TECHNOLOGY

Implementing the IEC Common Information Model for Distribution System Operators.

Risto-Matti Keski-Keturi

Examiner: Professor Pekka Verho
Thesis examiner and subject were approved in the Faculty of Computing and Electrical Engineering Council meeting 8th June 2011

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Electical Engineering
Risto-Matti Keski-Keturi: Implementing the IEC Common Information Model for Distribution System Operators
Masters Thesis, 91 pages, 2 appendix pages
November 2011
Subject: Electric Energy Technology
Examiner: Pekka Verho
Keywords: Common Information Model, Service Oriented Architecture, Distribution Management System

This thesis evaluates a move towards a Service Oriented Architecture design in ABB MicroSCADA Pro DMS 600, a Distribution Management System. The IEC standards 61968 and 61970, commonly referred to as the Common Information Model are expected to provide a foundation for building such a computing architecture in an interoperable fashion.

A Service Oriented Architecture differs in some ways from the existing DMS 600 architecture. In addition to identifying these differences, a short look is given to some possible changes in the business environment of both system vendors and operators made possible by the SOA. Security and other design issues raised by SOA were discussed.

The maturity of the CIM data types is evaluated by implementing a network data import functionality on the DMS 600 system using the CIM/RDF network model format. This part of the standard is found to be in a usable state for application in production environments, although additional CIM profiles would be useful.

The internal communication protocols of the DMS 600 system are found to be suitable for most types of information exchanges to be translated into such CIM messages. Some Service Oriented design principles are found potentially beneficial to the DMS 600 system internally, and proposals are given for some future development work.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Sähkötekniikan koulutusohjelma
Risto-Matti Keski-Keturi: IEC Common Information Modelin hyödyntäminen jakeluverkon haltijan tietojärjestelmissä
Diplomityö, 91 sivua, 2 liitesivua
Marraskuu 2011
Pääaine: Sähköenergiatekniikka
Tarkastaja: Pekka Verho
Avainsanat: Common Information Model, Palvelukeskeinen arkkitehtuuri, Käytöntukijärjestelmä

Tässä työssä tutkitaan palvelukeskeisen arkkitehtuurin ABB MicroSCADA Pro DMS 600 käytöntukijärjestelmälle tuomia mahdollisuuksia. IEC standardien 61968 ja 61970, joihin usein viitataan nimellä Common Information Model, odotetaan auttavan järjestelmien yhteensopivuudessa ja avaavan mahdollisuuden siirtyä kohti tällaista arkkitehtuuria.

Olemassaoleva DMS 600 sisäinen arkkitehtuuri ei vastaa täysin palvelukeskeisen arkkitehtuurin periaatteita. Nykyisen järjestelmän ja palvelukeskeisen arkkitehtuurin erojen tunnistamisen lisäksi työssä käsitellään mahdollisia liiketoiminnan muutoksia, joita palvelukeskeinen arkkitehtuuri saattaa aiheuttaa. Työssä käsitellään myös palvelukeskeisen arkkitehtuurin aiheuttamia uusia tietoturva- ja muita teknisiä kysymyksiä.

CIM tietomallin soveltuvuutta käyttöön tutkittiin toteuttamalla verkkotiedon tuontiominaisuus CIM/RDF-muodosta DMS 600 sisäiseen verkkomalliin. Verkkomallia käsittelevä standardin osa havaittiin nykyisellään pääosin riittävän yksiselitteiseksi myös DMS 600 tarpeisiin.

DMS 600 sisäisen viestiliikenteen havaittiin pääosin soveltuvan myös palvelukeskeiseen malliin muunnettavaksi. Jotkin palvelukeskeisen arkkitehtuurin suunnitteluperiaatteet saattavat olla positiivisia myös DMS 600 kehityksessä huomioitaviksi, ja työ antaakin joitain suosituksia tuotteen tulevalle suunnittelulle.

FOREWORD

As this thesis is finally closer to being printed, I feel both relieved and anxious to continue on the same subject. I believe the matters discussed in this thesis are interesting and can benefit everyone working within electricity distribution. I hope I can communicate the reasons for that later on in this text.

A thank you is in order for the thesis examiner, professor Pekka Verho. Not only for his input during the actual process of writing the thesis, but also for some interesting lectures on this very topic early on in my studies.

For my colleagues I am grateful for providing a working environment which I am happy to enter, even when it means leaving behind the freedom of student life. With this thesis, my instructor Ilkka Nikander has been especially helpful, and most of all, showing interest in the subject.

Lastly I want to thank my parents and my girlfriend for supporting me in my studies.

Tampere, November 22nd, 2011
Risto-Matti Keski-Keturi

CONTENTS

1. Introduction 1
2. A breakdown on the DSO business 3
2.1 Common types of computing systems 4
2.1.1 Distribution Management System 5
2.2 Development trends 8
2.2.1 Technological 9
2.2.2 Business 10
2.3 Current situation of interfaces 11
2.3.1 TCP/IP socket messages 12
2.3.2 COM/DCOM 13
2.3.3 Database sharing 13
2.3.4 File transfer-based data exchange 14
2.3.5 Web technologies 14
3. Service Oriented Architecture 17
3.1 Deploying SOA 18
3.2 Concepts 18
3.3 Software as a Service 20
3.4 Business perspective 20
3.4.1 System vendor 20
3.4.2 Network operator 21
3.5 Conclusion 22
4. Applicable data exchange standards 25
4.1 The CIM standards 25
4.1.1 The CIM model 26
4.1.2 Standard development 28
4.1.3 CIM interfaces 29
4.2 Other data exchange standards 29
4.2.1 Geographic data exchange standards 29
4.2.2 MultiSpeak 31
4.2.3 IEC 61850 32
5. Previous implementations of the CIM standards 33
5.1 In MicroSCADA pro DMS 600 33
5.1.1 An AMI interface to DMS600 33
5.2 Other vendors 35
5.3 Network operators 36
6. Implementation considerations 39
6.1 Encryption and signing 39

6.2 Denial of Service . . . 40
6.3 Data integrity . . . 42
6.3.1 Model Resource Identifiers . . . 43
6.3.2 Version control of network assets . . . 44
6.4 Performance . . . 46
6.5 Available technologies . . . 47
6.5.1 XML . . . 47
6.5.2 RDF and semantic markup . . . 49
6.5.3 Simple Object Access Protocol . . . 53
6.5.4 Service Discovery and metadata . . . 54
6.5.5 Web Service frameworks . . . 55
6.5.6 The communication bus . . . 56
7. Identification of services in DMS600 . . . 59
7.1 The internal architecture of DMS 600 . . . 59
7.2 Possible functionality to be implemented as services . . . 62
7.2.1 Processing AMR events . . . 62
7.2.2 Workgroup positioning . . . 63
7.2.3 Sharing outage data . . . 64
8. CIM network model format . . . 69
8.1 Topology . . . 69
8.1.1 Voltage levels . . . 71
8.1.2 Transformer modelling . . . 75
8.1.3 Auxiliary equipment and connectors . . . 76
8.1.4 Identifier consistency . . . 76
8.1.5 Coordinates . . . 77
8.1.6 Conductor Types . . . 79
8.2 Differences between transmission and distribution networks . . . 82
8.3 Implementation of network data import . . . 83
8.4 Model and profile maturity . . . 86
9. Conclusions . . . 89
9.1 Future work . . . 90
Bibliography . . . 92

TERMS

AMS	Asset Management System
AMI	Advanced Metering Infrastructure
AMR	Automated Meter Reading
AM/FM/GIS	Automated Mapping/Facilities Management/Geographical Information System
ASCII	American Standard Code for Information Interchange
CIMug	The CIM User Group
CIM	Common Information Model, defined by a set of IEC standards
CIS	Component Interface Specification
CIS	Customer Information System
(D)COM	(Distributed) Component Object Model
DBMS	Database Management System
DMS	Distribution Management System
DMS 600 NE	ABB MicroSCADA Pro DMS 600 Network Editor
DMS 600 WS	ABB MicroSCADA Pro DMS 600 Workstation
DSO	Distribution System Operator
EA	Enterprise Architech, a UML modeling software utilized by the CIM user group
EDXL	Emergency Data Exchange Language
EMS	Energy Management System
ESB	Enterprise Service Bus
FDIR	Fault Detection, Identification and Restoration
FTP	File Transfer Protocol
GIS	Geographical Information System
GML	Geography Markup Language
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Environment
IEC	International Electrotechnical Comission
IED	Intelligent Electronic Device
IP	Internet Protocol
JMS	Java Message Service
LV	Low Voltage
(OMA) MLP	Mobile Location Protocol
MSMQ	Microsoft Message Queuing
MV	Medium Voltage
NIS	Network Information System

NRECA	National Rural Electric Cooperative Association
OASIS	Organization for the Advancement of Structured Information Standards
OLE	Object Linking and Embedding
OMA	Open Mobile Alliance
OMG	Object Management Group
OPC	OLE for Process Control
OWL	Web Ontology Language
PLC	Powe Line Communication
RDF	Resource Description Framework
RSS	Really Simple Syndication
SaaS	Software as a Service
SCADA	Supervisory Control and Data Acquisition
SOA	Service Oriented Architechture
SOAP	Simple Object Access Protocol
SPICE	Simulation Program with Integrated Circuit Emphasis
SQL	Structured Query Language
SSL	Secure Sockets Layer
TAM	Telephone answering machine
TCP	Transmission Control Protocol
UBL	Universal Business Language
UCAIUG	UCA International Users Group
UML	Unified Modeling Language
W3C	World Wide Web Consortium
WS-*	The Web Services standard family
WSDL	Web Service Definition Language
XML	eXtensible Markup Language

1. INTRODUCTION

Service Oriented Architechture, or SOA, is a computer system design paradigm. As a concept, SOA may seem vaguely defined. Definitely it is not a single technology, although some technologies such as XML and Web Services are nowadays seen as a key part in realizing a SOA deployment.

The promise of SOA, like many software development trends before it, is to simplify deployment of new systems. The SOA way of integrating systems is creating services, with which different systems can interact. SOA discourages proprietary point-to-point interfaces in favor of a centrally managed repository of distributed services.

To enable these services to exchange data, they must use a commonly defined language. In terms of markup, web technologies such as XML act as the enabling technology. A commonly agreed way of expressing the information content is needed. When it comes to electricity transmission, distribution and generation, the IEC CIM standards family steps in. It is an effort to create guidelines for expressing the data of electricity network operators.

The goal is to make interfaces so transparent that a new component can be plugged in with only trivial configuration. While true plug-and-play capability will propably not be realized in the foreseeable future, the new ways of designing systems have a potential of lowering the costs of system integration.

If systems are able to talk to each other easily, the benefits are numerous. For the grid operators, vendor lock-in will be eased. Data is no longer captive in a proprietary system, only to be liberated by countless hours of consultant work. For vendors new possibilities arise. No longer do they need to offer a complete software stack to fill all the needs of every client. When system interaction is fluent, a company may be able to successfully market a piece of software to fulfill a single task of the customer.

Figure 1.1 shows a possible future system architechture of an electical network operator. The real-time needs are filled by corresponding standards from the IEC 61850 series. Business functions apply the data formats commonly adopted for such tasks, regardless of sector. The core non-realtime systems for network management use the CIM standards to talk to each other. Traditional systems, Distribution Management System (DMS), Customer Information System (CIS) and Asset Management (AM)

are overlaid on the picture, but they share functionality where traditional architectures would overlap. Data in different formats is converted using widely adopted XML technologies such as XSLT when it passes to another domain. A separate data warehouse is also depicted in the figure. It may be used for cross-checking data in individual systems and perhaps as the primary data storage for some.

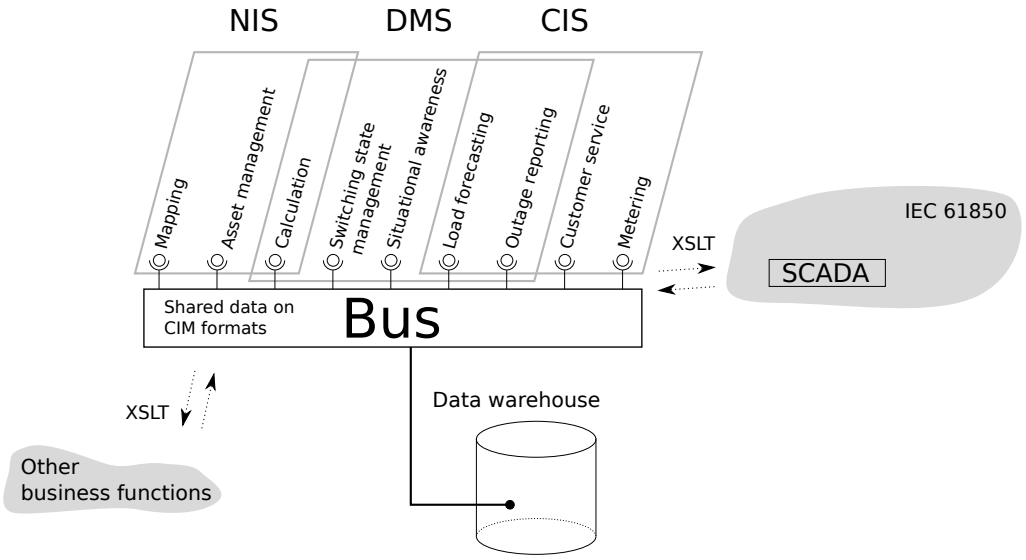

Figure 1.1: An illustration how a SOA system might look in the context of a Distribution Network Operators computing system, and how it relates to the traditional view of software packages, Network Information System, Distribution Management System and Customer Information System.

The bus shown may be just a catalog of pointers to services offered by different systems. It may also be an Enterprise Service Bus, a system which passes messages from system to another, verifying data, performing transformations and providing reliable caching and queuing while doing so.

The most promising part about the SOA paradign is that one does not need to convert all systems to it at once. The SOA principles are best kept in mind even when designing point-to-point links between systems. One day another piece of the SOA puzzle may be added, and at that point existing work can be leveraged. The typical maze of systems is a product of decades of additions and extensions. The SOA principles should be embraced when updating these legacy systems – one day the effort may pay back.

2. A BREAKDOWN ON THE DSO BUSINESS

Electrical networks can be clearly categorized in two distinct groups, distribution systems and transmission systems. Transmission systems are operated on large geographical area, usually on national level and often include connections across national borders. They are the backbone of the electricity network. Transmission systems operate on high voltage levels and have strict reliability requirements. High reliability is achieved using a meshed topology, where nodes are supplied by multiple connections in normal operation.

Distribution networks are the usually medium and low voltage networks that span a regional level. They are connected to the transmission system, and from the point of view of it are the energy consumers. A distribution network provides electricity to individual consumers in a cost-effective way. The networks are usually operated as a branched topology, although meshed connections may exist for redundancy during abnormal situations. The scope of this thesis is limited to the computer systems of the distribution network operators. Although some overlap exists in both software and operational procedures, the domains of transmission and distribution generally require different operational procedures.

Electricity networks are often seen as a natural monopoly due to high initial investment costs. In most countries this status is enforced by law, while in others the power system is actually owned by a public utility. In countries with private distribution system operators, they may have elaborate reporting requirements and price controls mandated by the law.

The operations of a DSO typically consist of distinct areas. Technical operations can be classified to network operation, planning, maintenance and construction. Customer service and billing are important non-technical aspects. Network planning is the ongoing effort to analyze future changes in the operating environment and direct investment accordingly to ensure operational goals are met in an economically sound manner. Maintenance consist of long-term condition monitoring of assets and timely replacement of components. Network construction is typically made up of clearly defined projects. Some construction tasks, such as laying cable or aerial lines happens more often while others, such as commissioning new substations is more rare and requires specialized know-how.

Network operation aims to provide a reliable system by optimizing the state of

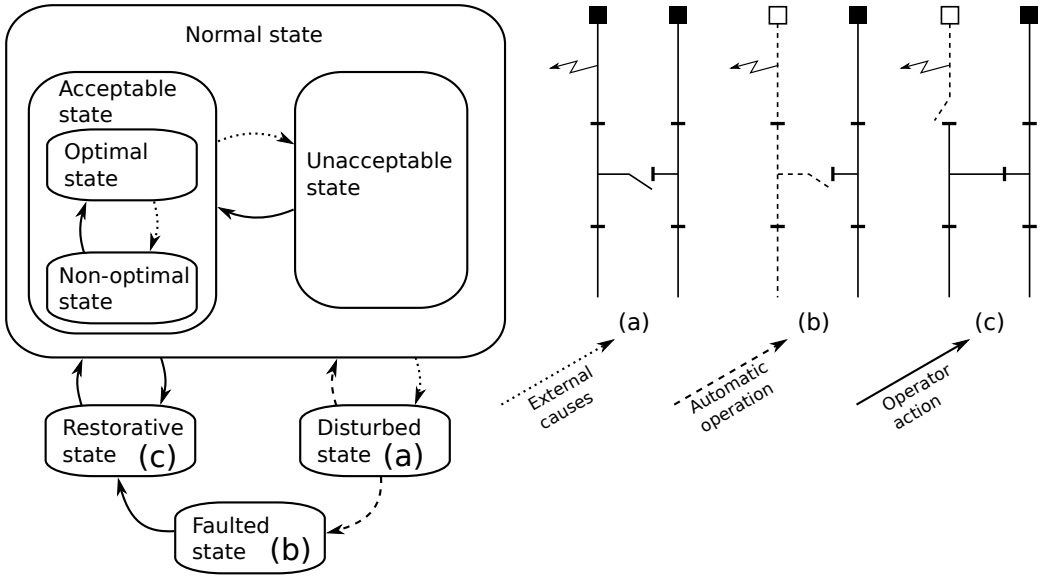

Figure 2.1: A typical fault isolation scenario on a radial distribution network. A fault occurs on one feeder (a). It is immediately disconnected by automatic operation of a circuit breaker (b). Supply is restored to as large part of the network as possible using a meshed interconnection that is normally operated in the open state (c). The state diagram was adapted from [1].

the network to best respond to unexpected situations, while ensuring losses in the network are as low as possible. The figure 2.1 shows a typical fault isolation scenario on a branched network topology with a mesh interconnection. Some networks use additional devices such as mid-line fault detectors or fuses to provide additional protection. Nonetheless, the main function of the operations team is to make the best decisions based on limited information in order to keep anomalies as isolated as possible. [1]

2.1 Common types of computing systems

A DSO typically operates multiple pieces of software from different vendors. It is hard to define the exact functional pieces of a type of software, but here we list some terms commonly applied to specific functionalities. This list only includes systems directly related to the operation of the power system.

A Supervisory Control and Data Acquisition system, or **SCADA**, is a computer software that communicates with measurement and control devices of the power system. SCADA software is also used in other fields, such as process control in plants and factories, or even HVAC and lighting functions in large buildings. A typical SCADA product is not specialized to only one function, but is versatile and can be deployed in very different environments.

SCADA interoperates with devices, which makes for strict security and perfor-

mance requirements. SCADA installations may even be deployed in a network completely isolated from the internet to ensure this.

Asset Management, or **AM**, is a database of assets that is used to track their state and plan long-term investment options. Other commonly used terms for similar system are AM/FM/GIS (Automated Mapping, Facilities Management, Geographic Information System) and NIS (Network Information System) which specifically includes functionality for network calculations. An AM may have some calculation and analysis tools for both planning and operation purposes. GIS in general refers to any system used for keeping a database where a geographical component is included and the term is commonly used outside the domain of electrical networks.

A Customer Information System,**CIS**, is a system for keeping track of customer contacts, such as billing. As billing is often affected by supply quality, a CIS may require data on the status of the network, yet it does not implement a network model. A related system, Trouble Call Management is used to track customer complaints to get information on the status of the network. This is especially important in installations that rely on medium voltage fuses for protection, as customer complaints may be in cases the only source of information of a fault.

Advanced Metering Infrastructure, abbreviated **AMI**, refers to deployment of remotely read or remotely controlled usage point eletricity meters. Another term associated with AMI when referring to it as a functionality rather than a system is AMR, or Automated Meter Reading. Although the primary goal of an AMI deployment is to ease billing, electrical network operation also benefits from the additional data such installations can provide. Depending on the system, AMI can provide real-time measurements and alarms in addition to historical energy measurement data. Future development is expected to allow AMI installations to operate as the customer interface to demand response on the micro scale.

2.1.1 Distribution Management System

Due to thesis concentrating on a Distribution Management System, it warrants a separate introductory section. The DMS is a system combining data from SCADA, network calculation engines and asset management to present a real-time view of the distribution network to the contol room pesonnel, and to record operational activity for later review. The DMS naturally takes a role of the information hub in a system, due to it requiring data from many separate systems. This means a DMS has to deal with many types of data exchange interfaces, leading to practical problems on implementation level, but also a great deal of possibilities for improving the overall experience of the control room and other stakeholders.

The main interface of ABB MicroSCADA Pro DMS 600 is a geographical view of the network laid on top of a map or aerial photographs. Colors are used to display

information of the state of the network to control-room personnel. In addition to topology analysis for determining unsupplied parts, DMS 600 includes information on load levels, voltage drops and protection status of the network. It also includes some analysis features to search for optimal operating state, such as selecting a disconnector configuration that minimises losses. Advanced planning features have been developed to allow for estimating life-cycle costs of network construction by taking expected outage costs of proposed configurations into account.

During normal operation, the DMS 600 warns the user when entering potentially dangerous switching states, such as having a short-circuit current on part of the network too low for relays to detect faults, or opening disconnectors that cannot handle the real-time load-current. The main view may be overlaid with additional information, such as real-time positioning of work groups.

During faults, the DMS 600 combines data from asset management and SCADA to find potential fault locations based on load- and fault current and impedance of the network, adjusted for the actual switching state and load level during the fault. It can advise the operator on propable fault locations based on these calculation. The main screen of the MicroSCADA Pro DMS 600 is shown in figure 2.2.

The operations tracking allows the DMS 600 to simulate any past switching state to allow for later review. From this data, reports can be generated for regulatory or internal review of operational performance of the DSO. Real time situational data can also be exported to other systems to give customers or other parties information on the network supply status.

The MicroSCADA DMS 600 system architechture includes clients running on workstation machines and exchanging data through both a proprietary TCP protocol and a database connection. A typical environment is illustrated in figure 2.3. The client side includes two separate applications: the Network Editor (NE), used for editing the network data and the Workstation (WS), the main program used for monitoring the network state.

The NE uses the database as the primary storage, but it is also responsible of writing a binary network file, a compressed representation of the network data for use by the WS instances. The creation of the binary network file requires user interaction, and is usually done after an operator performs modifications to the network.

The instances of the WS application load the static network data file as well as background maps using standard operating system file sharing protocols. When an operator performs changes that need to be propagated to all instances, usually data is written on the database and a DMSSocket message is sent. Other instances then know to refresh their data from the database.

Most of the calculations are performed on client side instances of the WS appli-

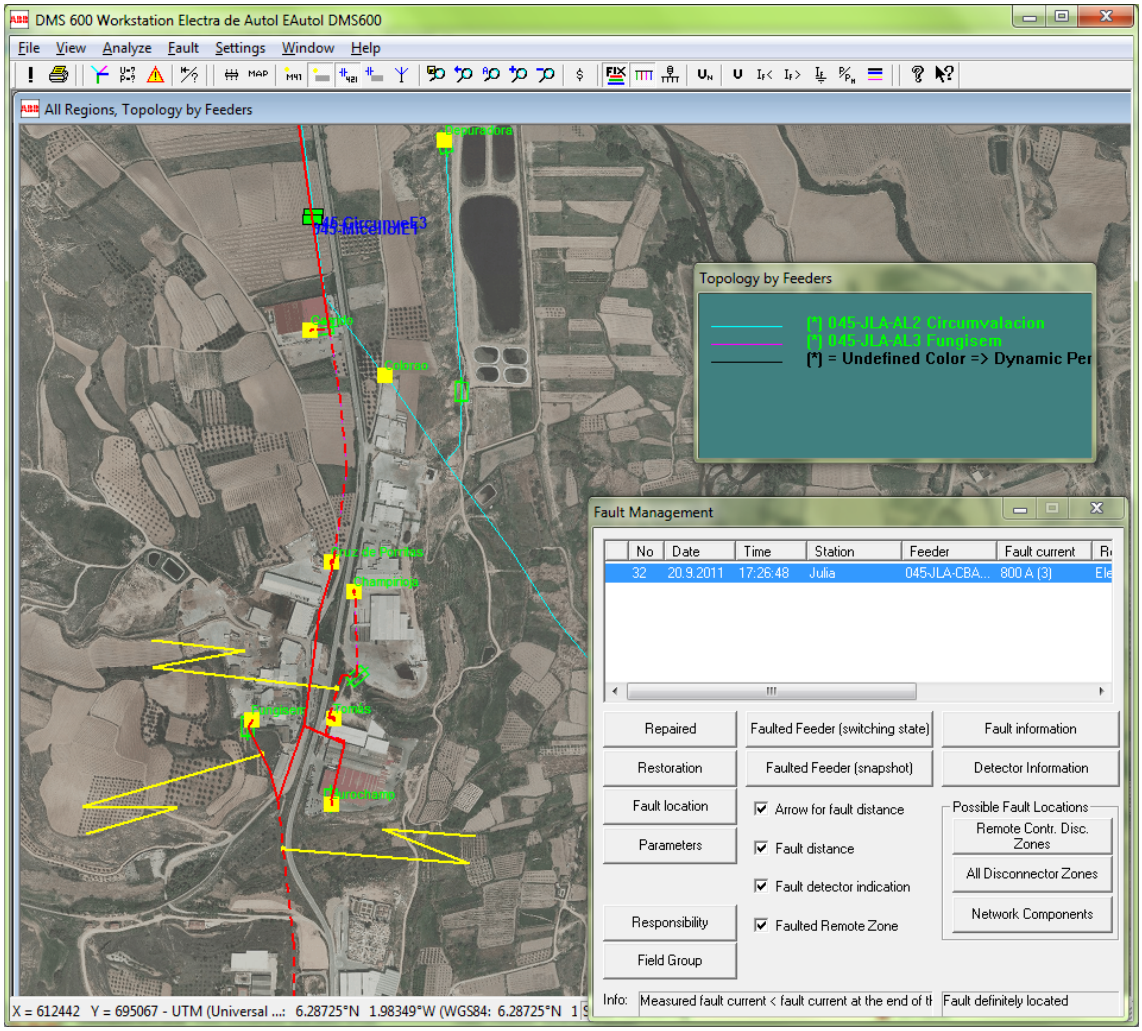

Figure 2.2: ABB MicroSCADA Pro DMS 600 displaying possible fault locations, overlaid on top of aerial photogaphs.

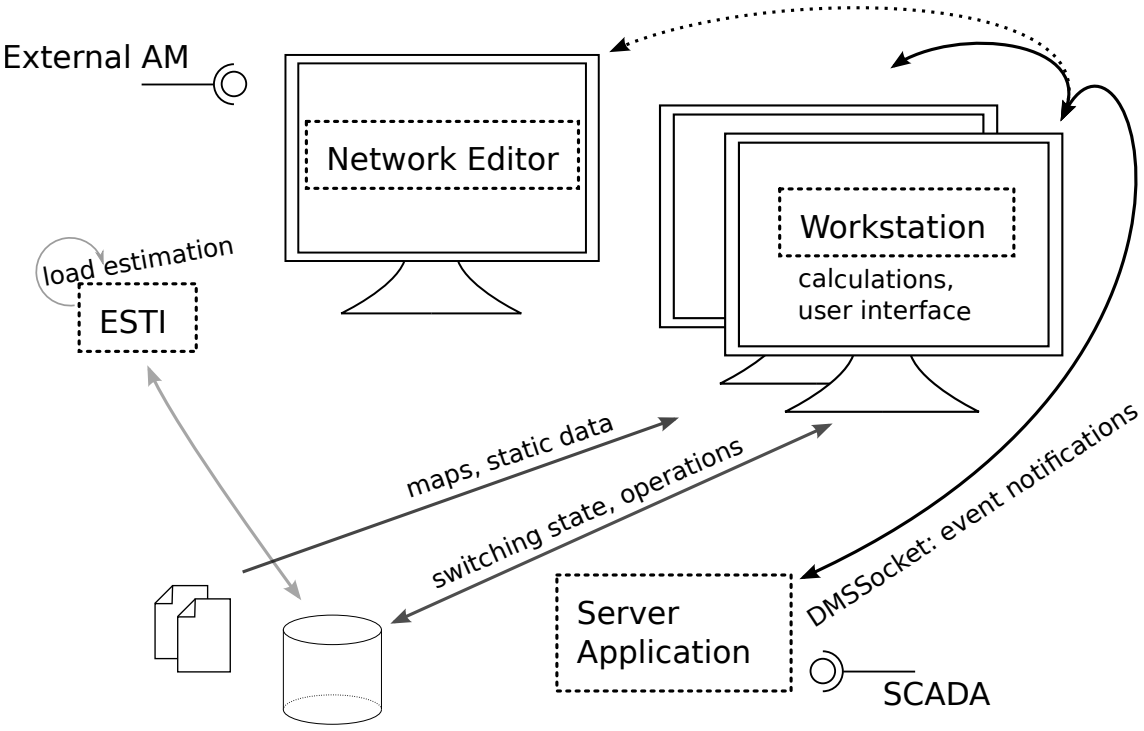

Figure 2.3: ABB MicroSCADA Pro DMS 600 environment.

cation. This means some calculation effort is duplicated on environments running multiple instances. For others, e.g. load estimation, a separate calculation engine is invoked periodically. The WS instances then load the data produced by this engine.

A single Server Application instance runs in the systems. The Server Application is responsible for tracking changes in switching state originating from a SCADA system. The SA communicates with WS instances through DMSSocket protocol.

2.2 Development trends

Historically, the distribution networks were often operated by an electrical utility that also operated electricity generation. Nowadays electricity distibution and generation are often decoupled into separate companies or at least departments within a company. Many countries enforce this decoupling by regulation. In Finland such law entered into force in 1995. This law defined price controls, mandatory reimbursements in case of a failure to provide realible service and required electricity sales and distribution business units to have separate accounting. [2]

Current environmental concerns, such as the effort to reduce greenhouse gas emissions have ripple effects on the electricity distribution industry. While electricity distribution does not cause emissions as such, distribution companies may be the only party capable of providing information or control required by other parties. As for energy losses in electricity distribution, methods for optimizing switching

states are already in place in DMS systems, but further increases in electricity cost - driven either by emission costs or fuel costs - may justify additional investment in the network itself.

2.2.1 Technological

Electricity distribution has been a rather static industry from a technological point of view since it appeared a century ago. From early on, the networks were mainly operated as alternating current systems that utilized different voltage levels to reduce losses. A modern transformer, the enabling technology of large scale energy distribution is still fundamentally very similar to the original invention.

Progress has been made in operating the networks more reliably. Circuit breakers controlled by electronic devices are increasingly accurate due to increased processing power and better sensors, allowing much more fine-grained constraints on protection. Development of hot-wire working techniques allow many types of maintenance work to be done without causing an outage on the network. Remote monitoring and remote control has enabled networks to operate with ever decreasing labor requirements. Modern computer systems provide ever improving visualization and situational awareness tools for operators.

A recent trend is the transformation to so called smart grid technologies. This means having huge numbers of additional measurements from the network and using these to operate the network more safely and reliably. The transition is fuelled by higher energy efficiency requirements, and programs such as the european aim of producing 20% of all energy with renewable power by the year 2020, which in turn push for more distributed generation. Smart grid technologies such as Automated Meter Reading take remote control and monitoring down to the level of individual customers. All new data must be processed in a meaningful way, and therefore there is a lot of pressure on software vendors to implement new functionality made possible by this additional data.

Safe operation of distributed generation also requires a lot more data on the state of the network compared to established ways of operating the network. Small generation scattered throughout the network result in the networks to be operated in a partial meshed topology, giving distribution networks more of the characteristics typically thought to only apply to transmission networks. Local generation also theoretically enables island operation, or keeping the network electrified even in the case of a fault cutting off upstream power input.

Demand response capabilities are starting to appear in AMR systems. The use of these is controversial, at least until electricity distribution contracts start including clauses that give the customers a significant incentive to allow parts of their non-critical loads, such as air conditioning, be remotely controlled. New possibilities for

load control may arise as new types of devices are acquired by customers.

One aspect of the smart grid trend is a tendency towards distributed intelligence. If Intelligent Electronic Devices[1] take over some of the decision making, response to transient situations is expected to be faster. Distributing the control process requires more information such as topology and state of other nodes to be sent down to individual control components. These components also have to be interconnected.[3] This development of distributed intelligence leads to considerations about transmitting the decisions and data they are based on back to the central system, to ensure control room situational awareness.

The influx of new technological capabilities brings forth concern about the cyber security of the system. Security threats may endanger both the safety and reliability of the system, but also the privacy of customers. An attack on the system that can be carried out remotely and with little threat of getting caught may open new motives for criminals. The exact measurements of individual load points may allow an attacker to determine facts about a specific customer – something that has never before been a concern related to power system operators. Legal liabilities of the power system operators in case of a security breach are also unclear as the issues are very recent.

2.2.2 Business

One clear trend in the business environment is the decoupling of grid maintenance and construction from the core business of the grid operator. This includes grid operators selling maintenance services to their competitors and also separate companies specializing in the maintenance of infrastructure, without owning any distribution network of their own.

The field of electricity network operation has some characteristics that make it a good candidate for outsourcing. An important aspect is the highly variable resource requirements during normal operation compared to emergency situations, such as a storm causing multiple faults in the network. Being a natural monopoly may actually help in advancing outsourcing. Where strict goverment regulations are in place, companies must already do cross-allocation of costs to separate business functions in their accounting. This means the first step towards outsourcing, identifying costs according to business units, may have to be taken just to comply with regulations.

Computing systems play a role in this process. In a 2009 questionnaire directed to finnish electricity network operators, incompatibility of computer systems was seen as a major reason against increased outsourcing. The specific reasons for this were not clear, but two main considerations are giving access to specific data without

[1] IED

compromising the whole system and transferring data between systems where a contractor does not posses systems from the same vendor. [4]

A related trend seen in various fields of business is consolidation to big units. Some smaller utilities are responding to the efficiency gains of larger operators by sharing parts of their operation in an umbrella company. Especially the latter causes interesting technical requirements, as a shared system may see multiple independent companies accessing it, but unwilling to share all data with others.

Going even further, some companies have taken a very passive role in actually operating the network. Some networks may have two separate business entities involved: the operator of the network and the owner. According to the previously mentioned questionnaire [4], companies see network planning as the most integral part of their business, a core competency.

2.3 Current situation of interfaces

The software stack at a typical electrical utility company contains an increasing number of applications, many from different vendors. The critical and more established systems were reviewed in the previous chapter. From a network operations perspective, an asset management software and a SCADA are central. A Distribution Management System unifies information from both of these to give operators a real-time view of the power system state. Multiple different calculation and analysis applications may have been built on top of the stack. From a business perspective the customer information system and any supporting billing software are invaluable, and as such there is a demand to interface these systems as well.

In addition to these, a smarter approach to network management calls for remotely read electricity meters, real time work group positioning and any information useful in predicting electricity demand. Some of this information may be produced externally to the DSO, for example weather forecasts may play a role in demand estimation.

Extensions such as automated fault detection, identification and restoration[2] are added on top of the SCADA system, sometimes by different vendor. New intelligent electrical devices perform more complicated tasks and therefore require more information from other systems. Different vendors may have different views as to the purpose of each system, which may result in overlapping features.

Figure 2.4 depicts some of the most typical and relevant interfaces available to ABB DMS600 today. Many more are to be found within the utilities' networks – shown are only some of those systems with direct interfaces to the DMS 600.

As the number of systems grows, the number of interfaces grows exponentially.

[2] FDIR

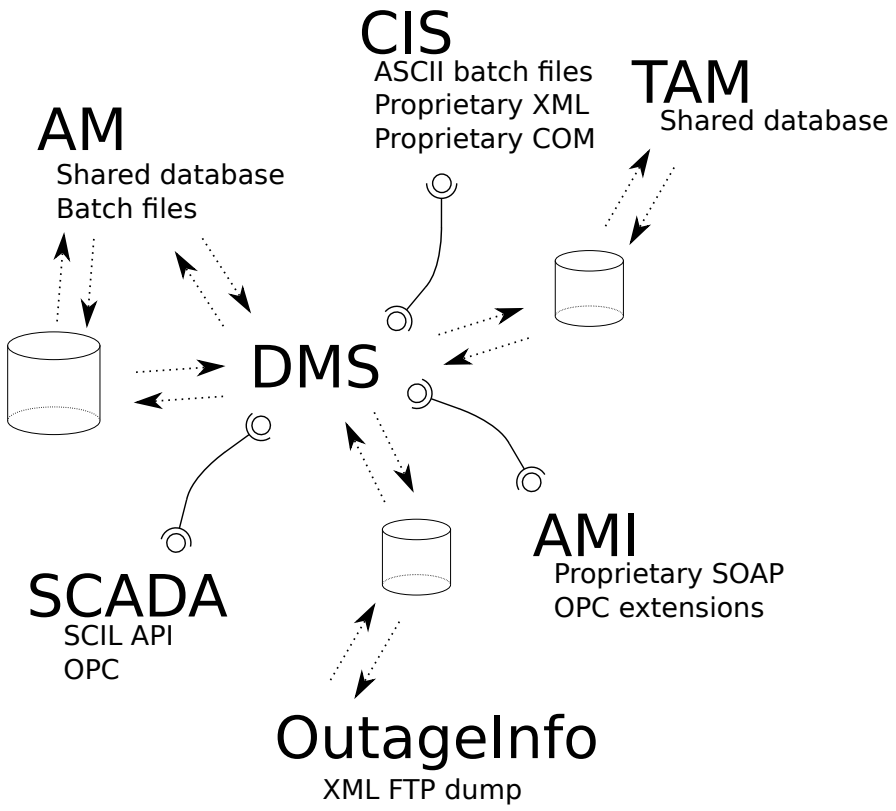

Figure 2.4: A subset of interfaces available to ABB DMS600.

Standards may be lagging behind implementations and a very typical operations room network carries multiple proprietary protocols, at times even specific to a single utility company. Very expensive development work may be required to get different systems to talk to each other just because a specific combination of software versions is not used anywhere else. To make matters worse, the designs may follow the best software development practices of multiple different decades. Documentation, where it exists and is available, may be severely outdated. A simple configuration change may require the involvement of an external consultant. Small changes may also result in non-obvious security implications.

2.3.1 TCP/IP socket messages

TCP/IP is the basic communication protocol nowadays used under pretty much all network implementations. Building interfaces based only on TCP/IP gives best possible control over what can be done and how, but it is left to the implementation to supply functionality such as security, data definitions and the like. All other interface types listed in this chapter add an abstraction layer on top of TCP/IP.

DMSSocket is a name used internally at ABB to refer to a proprietary, light-weight protocol built on TCP/IP socket messages to handle synchronization and small data transfer tasks in the DMS600 environment. It actually consists of two

separate protocols, one (Unknown socket) based on pure ASCII messages and the other on proprietary binary messages.

The internal protocol mainly sends notifications of events in the system, instructing the recipient system to refresh data from a shared source, such as the database. Usually the data carried in the socket messages is insufficient to update program status.

The ASCII protocol provides less functionality, but is designed for very simple interfacing from third-party systems. Currently it exposes such features as AMR event input to the DSM600 system, desktop integration between the DMS600 Workstation and third party customer service software and receiving work group positioning information from external systems.

The DMSSocket protocols are lightweight and simple. Security is achieved by isolating the system from untrusted networks: as such, the DMSSocket protocol can only be safely exposed to computers in a tightly regulated internal network.

2.3.2 COM/DCOM

(Distributed) Component Object Model is a proprietary Microsoft interface for interprocess communication on the Windows operating system. Although gradually superceded with .NET technologies, it is still a widely used API[3] for data transfers.

In DMS600, COM is used among other things for communication between the DMS and ABB MicroSCADA and for making requests to open specific user interface features from separate programs in the DMS workstation. OLE for Process Control, or OPC is a widely deployed interface standard to SCADA systems, and was originally built on DCOM technology. Future revisions of the standard are expected to migrate from DCOM to Web Services technologies.

2.3.3 Database sharing

In a database sharing model two or more applications connect to a shared database. Usually it is well defined in the interface specification as to where each application writes data and what data it consumes. This approach works for best when one vendor specifies the data formats and others implement very specific transactions.

The most obvious problem is that comprehensive access restrictions may be difficult or, depending on the database vendor, impossible to define. This may lead to data conflicts or even vulnerabilities, if too broad access rights are granted. The problem is compounded by the fact that sotware vendors are rarely in the position to unilaterally standardise on only one supported DBMS vendor.

The DMS600 suite heavily leverages database sharing for internal communication,

[3]Application Programming Interface

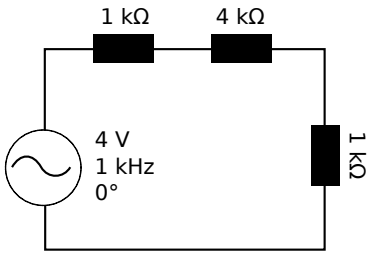

Figure 2.5: An example electronic circuit, described in the SPICE format on code listing 1.

a situation where database sharing arguably is appropriate. Access restrictions not supported by the database engine are partially built into the software. Some external interfaces exist in DMS 600 leveraging database sharing, among them an interface to an external automated telephone answering machine system.

2.3.4 File transfer-based data exchange

One way of creating communication interfaces between applications is based on transferring files. This method allows the usage of multiple different protocols, such as FTP[4] or the networking capabilities built on Microsoft Windows, depending on the environment in use. Current ESB suites often allow adapters for file-based transfer standards to provide routing and transformation for such older interfaces.

File-transfer based interfaces can further be categorized in two distinct parts. More recent file types are often defined as XML schemas, allowing the use of third party software to aid in the file processing. Using XML also simplifies processing the file in an ESB. Simply using XML however does not make a file format definition a standard or allow for interoperability.

Older interfaces may use proprietary ASCII[5] text files, or binary files. An ASCII file contains human-readable characters but may or may not provide structure. An example of an ASCII based format can be found on figure 2.5 and code listing 1.

A binary file contains data where e.g. numeric types are filled in using their internal memory representation. A binary file is very difficult to parse without knowledge of the file format, and even in that case existing software may not be able to parse it without implementing a specific adapter.

2.3.5 Web technologies

Many standards and de facto standards have risen with the popularity of the internet. These standards include SOAP, Web Services and others. XML is heavily leveraged throughout. Especially so-called WS-* standards are useful as starting

[4]File Transfer Protocol
[5]American Standard Code for Information Interchange

Code listing 1 An input file for SPICE, an established electrical simulation program showing a typical ASCII-based file type. The circuit in questiong can be seen in figure 2.5.

```
R3 V3 0 1k
R2 v2 v3 4k
R1 v1 v2 1k

VVin v1 0 SIN(0 4 1k 0 0)

.end
```

points for new interfaces. They extend existing protocols, such as TCP/IP and HTTP[6]. Many of them are published by the same W3C[7] responsible for the ubiquitos internet protocols and concepts.

While a WS-* type implementation tries to enforce an up-to-date documentation in the form of XML schemas and WSDL[8], an adhering interface does not automatically equal to a standard interface. Data types are often still proprietary and if bad development practices have been in place, the interface may for all practical purposes be just as opaque as a decade-old binary transfer file.

In the end the WS-* type implementations remain the modern choice. A SOA deployment would naturally use these technologies, and existing implementations are easy to upgrade as long as adequate documentation is available. In the case of ABB DMS600, the WS-* technologies are the preferred way of creating new interfaces, and recent additions include AMR deployments and workgroup positioning interfaces.

Most common IDEs provide a variety of tools to easy the development on top of these technologies. On the flipside supporting a recent revision of a WS-* standard may be hindered by lacking support of the development tool of choice. Fragmentation remains a problem, with multiple standards proposed with overlapping functionality.

[6]Hypertext Transfer Protocol
[7]World Wide Web Consortium
[8]Web Service Definition Language

3. SERVICE ORIENTED ARCHITECTURE

As requirements for automated data processing grow, new functionalities need to be implemented. Often these new functionalities are provided as separate systems, possibly from vendors not yet delivering products to the computing environment of a company. The trend of increasing number of systems from different vendors interoperating with each other creates an ever greater number of interfaces required between them. If all systems have to be interconnected, the number of interconnections will become $N(N-1)/2$, where N is the number of systems. Even assuming limited interconnections, the maintenance requirements of these interfaces will saturate new development at some point.

To mitigate this problem shared by all organizations operating a complex computing system, the design guideline of Service Oriented Architecture has emerged. SOA is not a single technology nor a standard. It merely refers to the principle of dividing software to granular blocks of functionality and sharing processing workload by means of mutually agreed interface protocols. Interfaces in a SOA deployment are loosely-coupled, which means they are implemented by clearly described protocols instead of ad-hoc interfaces.

While at first sight dividing existing systems to even smaller pieces of functionality - *services* - might seem to increase the maintenance workload, the idea is that well-defined protocols allow different systems to reuse the processing capabilities exposed by other systems. Ideally, to implement new functionality, only the novel part has to be created from scratch. Say a new algorithm of solving power flow is discovered (or, more realistically, made practical by advances in computing technology) to implement it a vendor only needs to refer to the interfaces for fetching a power system model from existing applications, which are preferably shared by multiple vendors, and actually only implementing the algorithm - instead of creating a set of tools to model the power system from scratch and fetching real-time data by means of a multitude of separate interfaces.

The key to a successful SOA deployment is standardization. Without it, SOA will only bring more interoperability hurdles. The standardization of the basic protocols is well under way and is agreed across different business domains. A key technology here is XML, which will receive a more in-depth review later. The other part of standardization are the interfaces within a business domain, such as power system

control. An international effort to reach such a standard is ongoing in the form of the IEC Common Information Model. [5]

3.1 Deploying SOA

SOA being a radical change in software development practices, most proponents agree that an all-out SOA rollout is undesirable. Instead, existing systems should be gradually SOA enabled. New deployments should blend in with existing operating procedures, yet all levels of an organisation should be involved in defining the optimal procedures into which computer systems should adapt.

The steps in the migration process are roughly

- Definition of data types used in the industry and within a specific organization
- Definition of business processes
- Definition of services required to model said processes
- Identification of existing implementations of said services
- Migration of said implementations to allow loose-coupling with the SOA infrastructure
- Implementation of additional services needed in the business process
- Management of the resulting architecture

Typically literature on SOA calls for the definition of the business data, required services and processes from the ground up, instead of being dictated by software vendors. For power system operators, the definition of data models is a done in cooperation with multiple software vendors in the form of the CIM standards. A general overview of the process of SOA migration and the development of the CIM standards is shown in figure 3.1. [6][7][8][9]

3.2 Concepts

This section explains some of the terminology associated with SOA. Technological and implementation details will be discussed in later chapters.

Service A service is a function exposed in the system. It ideally provides an interface to do a single task. Applications connect to the service using a predefined protocol. A service may invoke other services to perform its task.

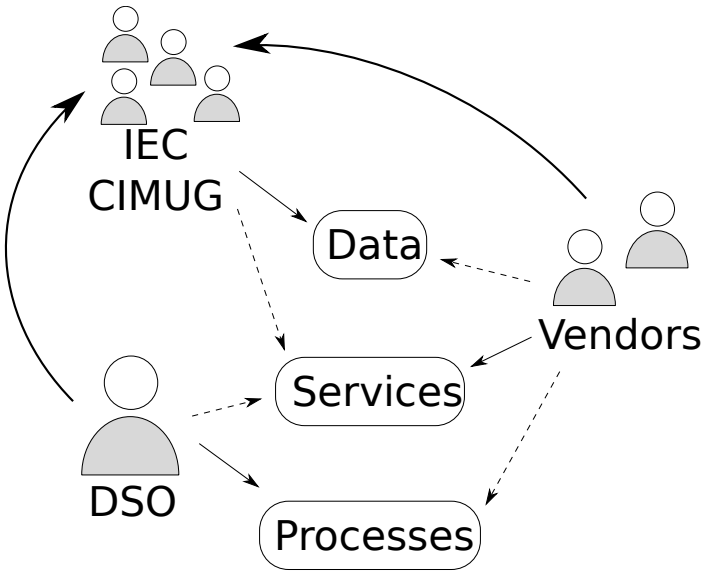

Figure 3.1: This figure shows a simplified model for the steps in migrating towards a SOA infrastructure and highlights the role of different actors in the field of power system operation.

Orchestration An orchestration is a workflow involving multiple service invocations. An example could be an orchestration for handling an incoming customer trouble ticket. This example orchestration would possibly invoke a service that handles notifying control room personnel of the problem and another that creates a data row in an outage statistics database.

Loose coupling Loose coupling[1] refers to connecting software using a well defined, open protocol instead of code-level invocations of program assemblies. Loose coupling ensures that a program does not care whether the services it invokes reside on the same system or over a network, or whether they are running on the same operating system, or indeed the operating system the service was originally deployed on.

Enterprise Service Bus An ESB is a platform for routing service requests in the network. An EBS typically executes the orchestrations. Features such as service discovery are usually a component of an ESB.

Representational State Transfer REST[10, ch 5] is a related but not interdependent concept to SOA. If an interface is RESTful, the state of an application is held only on client side. Requests to a RESTful service hold all necessary data to perform the request, whereas in a stateful interface the server must keep track of client applications and their state.

[1] as in *loosely-coupled interface*

3.3 Software as a Service

Software as a Service[2] refers to an emerging business model where software is licensed less as 'goods', as is the case with currently common perpetual licences, but more as a service. In a SaaS model a customer pays for software based on their usage instead of in large chunks of purchasing licences and upgrades. To achieve this, the software is run and the data is hosted on the providers system.

The benefits of SaaS are faster feature delivery (because only one copy of the software is installed, at the vendors system) and more adaptable pricing structure. Points of concern, especially for software used to run basic infrastructure such as power systems, are data security and privacy. When data is hosted by the service provider, the service provider should be under the same data retention legislation as the DSOs.

SaaS should not be confused with SOA. The term *service* in these abbreviations refers to a different concept - in SOA a software development concept, and in SaaS a software distribution concept. However, a mature SOA deployment opens possibilities for moving parts of the system to a SaaS infrastructure. This creates new revenue generating opportunities for vendors. [11] [12]

3.4 Business perspective

A quick look from the business perspective is in order to analyse the basic changes in the operating environment of parties involved in moving towards a SOA design. This chapter is divided to both the system vendors and the network operators, eg. the clients of a system vendor. A deeper analysis is beyond the scope of this thesis, so this chapter will be limited to a superficial listing of possible scenarios.

The assumption here is that multiple vendors start offering systems that expose parts of their functionality as standards-compliant services. The economic benefits of SOA will only be fully realized if many vendors embrace them.

3.4.1 System vendor

The primary gain for a system vendor is to ease development of new features by making existing functionality easily accessible to new modules. The assumption is that in a mature SOA ecosystem many basic building blocks already exist and the need for reimplementation is lower. This of course should make development cheaper and faster.

The benefits to completely new entrants seem enticing. Even only developing a single service, be it an algorithm, database or user interface, it would be possible to

[2] SaaS

offer a usable product in an environment where the required software infrastructure is easily interoperable. Possibly a SOA ecosystem would ease the path from a limited-scope research project into an actual product.

For established vendors who already posses a lot of the basic functionalities needed to introduce new features, the benefits seem lower than completely new entrants. A completely closed product would allow a vendor to raise the entry costs for new competitors. In business environment where standards are mature enough to support deploying service oriented architectures, established vendors would either have to choose competing with a closed product and providing all functionalities for a complete system, or embrace the new environment and start focusing on core capabilities while allowing some features to be provided by competitors.

In this environment of mixing small parts of functionality into an usable system, new markets may arise for consultants to actually ensure the services cooperate with each other and that the infrastructure (ESB, service discovery, network) is stable. From the point of view of the current market, the vendor that delivers the main package of the environment may be in a good bidding position to take on the role of this integrator. Therefore, acquiring the resources to maintain a service environment may become a good strategy for vendors such as ABB, if it is decided that this business segment is desirable.

Marketing and pricing of existing systems may have to be adjusted to respond to an environment where purchases are done for only a small fraction of the complete feature set offered by the product. Customers are in danger of ending up paying many times over for the same functionality when overlapping systems are deployed, and this point should be leveraged in the marketing of products that can avoid overlapping features. In the bidding process, a vendor that can clearly offer only what is asked should be able to give the best offer.

Instead of all-out deployments for a total replacement of existing systems, customer relations could build from small-scale projects to probe the capabilities and willingness to cooperate of vendors. The bidding process could possibly move from making specifications to doing more actual engineering work as proof-of-concept.

3.4.2 Network operator

A power system operator is in this business the customer. For them, in principle a SOA future should bring more flexibility in selecting the best products from all vendors for each task, when there is less need to worry about interoperability. In any case, modernization of the power systems towards the smart grid goals will introduce multiple additional systems to their environment, so interoperability will become increasingly important.

All new software purchases should from the start of the bidding process take

into account the new possibilities. Functionalities and price alone should not be the deciding factor in a purchase, instead more emphasis should be given to assuring all systems are interoperable and also expose their current functionalities sufficiently for new features to be added by third parties after the commissioning of the system. System operators should start gathering the required know-how to make successful purchases that fulfil these requirements.

Another factor driving the system operators towards the SOA principles is increased outsourcing of their business. This was the driving motivation behind the LIPA case discussed in chapter 5.3. A computing environment where SOA principles are followed ensures that when a provider for certain function changes, the new provider can use their own systems instead of doing a costly adaptation to whatever environment was in place by the previous contractor.

A key decision is whether system operators want to keep the knowledge of the standards in-house or start using consultants from early on in acquiring new software. Clearly the task of tracking standards in the fields of both power systems and IT, and ensuring compatibility with them during the bidding process, is not a simple one. Some companies, such as LIPA clearly take an active role in building their system. A vendor, however, would be wise not to assume this will happen with all DSOs. The ease of purchasing a complete system may be enticing enough for the DSOs to discourage from researching the SOA technologies themselves. In this case, a vendor that can offer a full package while combining best available technology from multiple other vendors, possibly also competitors, may be in a good position to gain new customers.

3.5 Conclusion

If standardization keeps advancing and truly SOA compatible products become available, the business environment may see some changes. It is important for both vendors and system operators to track the state of standards and make sure their processes take the new possibilities into account.

The vendors must be able to give definite answers on their compatibility level with emerging standards and their customers should have enough expertise to ask the questions. If SOA really catches on, established vendors may find themselves in a position where they have to decide whether to invest in being compatible or rely on existing momentum to take out competition.

From the point-of-view of ABB, the decision to embrace standards is clear and written down as integral to company strategy. When it comes to CIM and SOA, it is not yet certain whether the technology is mature enough to be deployed in production. In anticipation, the answer to some questions should be made clear: Are we capable of

- delivering a small feature to be integrated on an existing system?
- making a competitive quotation on said feature and is our pricing structure adapted to such sales?
- integrating third party software to fill gaps in our product?
- offering a partial feature set and sufficient standard interfaces to respond to specifications we cannot fully deliver alone?

Additionally, for the project organisation the question is whether we are comfortable of becoming the system integrator in projects with multiple vendors involved. A positive answer depends on whether the business prospect is considered lucrative and whether there are enough resources to support the infrastructure required in such projects. This thesis will further attempt to identify the technologies required for successful participation in the market as the system integrator.

4. APPLICABLE DATA EXCHANGE STANDARDS

This chapter explores some of the standards applicable to a DMS application. The standard mainly of interest is the IEC Common Information Model (CIM), but for tasks where existing inter-industry standards exist, those should propably take precedence in implementations unless the CIM provides additional benefits, as this eases interfacing with systems developed for other uses later on.

4.1 The CIM standards

The Common Information Model is a set of standards published by the International Electrotechnical Commission. The standards in question are the IEC 61970, *Energy management system application program interfaces (EMS-API)* and IEC 61968, *System interfaces for distribution management (SIDM)*. IEC 61968 is an extension for distribution management, sharing the majority of its data types with IEC 61970, while adding some minor details to network models and many message types for other related functions needed in distribution system operation. A complete list of published and planned CIM standards is compiled in appendix 1.

The IEC structure is made up of Technical Committees, which in turn contain many Working Groups. In the case of CIM, the Technical Commission in charge is TC 57, *Power system management and associated information exchanges*, which also publishes such standards as the IEC-61850, concerning communication between IED:s and SCADA systems, titled *Communication networks and systems for power utility automation*. The working groups dedicated to the CIM model are WG 13, in charge of IEC 61970, and WG 14 for IEC 61968.

In addition to the official IEC structure, a non-profit organization called UCA International Users Group is dedicated to facilitating interoperability for products using the IEC CIM. Vendors and end users, namely system operators, can join UCAIUG. The organization influences the development of the CIM standards and holds periodical interoperability tests. UCAIUG is further divided to groups, the most relevant in this case being the CIM User Group, further abbreviated CIMUG.

4.1.1 The CIM model

According to CIMUG, the CIM model is

> an abstract model that may be used as a reference, a category scheme of labels (data names) and applied meanings of data definitions, database design, and a definition of the structure and vocabulary of message schemas. The CIM also includes a set of services for exchanging data called the Generic Interface Definition (GID). Due to its abstract nature, the CIM may be implemented in many ways. Some users may want to be more specific to support a particular product or project; others may want to limit its scope. The abstract nature of the CIM leads to flexibility and competing demands for its enhancement. It is important to remember that the CIM does not have to be implemented in its entirety in a given project.

The model is built on object-oriented design philosophy. It is made up of classes, which reference each other by inheritance and association. If a class inherits another, it includes all properties of the parent class. A reference means that the data type of one property of one class is an instance of another. An instance of a class is usually called an object.

For example, refer to figure 4.1. An *OutageRecord* describes an outage in the network. It is a subclass of *Document*, which includes information such as the creation and modification dates of the document. *Document* itself is a subclass of *IdentifiedObject* which serves to give a unique identifier to all objects within a domain, such as the computing environment of a single distribution system operator. In addition to the inheritance chain, an *OutageRecord* associates with zero or more *OutageStep*s, which is used to list the equipment that was affected by the outage.

The CIM standards are very vague on implementation details. Communication protocols and markup standards are left up to the implementation to decide, although XML schemas are provided as non-normative appendices of some of the standards. For network model data transfer, an RDF schema is provided.

For protocols, CIM standards use an abstract verb/noun structure. A message exchange could be described for example as a client initiating a request to *CREATE OutageRecord*, where *CREATE* is the verb and *OutageRecord* is the noun. The actual communication exchange would involve wrapping this abstract command into for example SOAP calls, depending on the implementation.

The model has many free-text properties. When using these, the semantics are no longer easily deduced from the standards. For example, an *IdentifiedObject* can be associated with a *Name* object, which gives it an alias name. The *Name* in turn references *NameType*, which is an object shared by many *Name*s within a

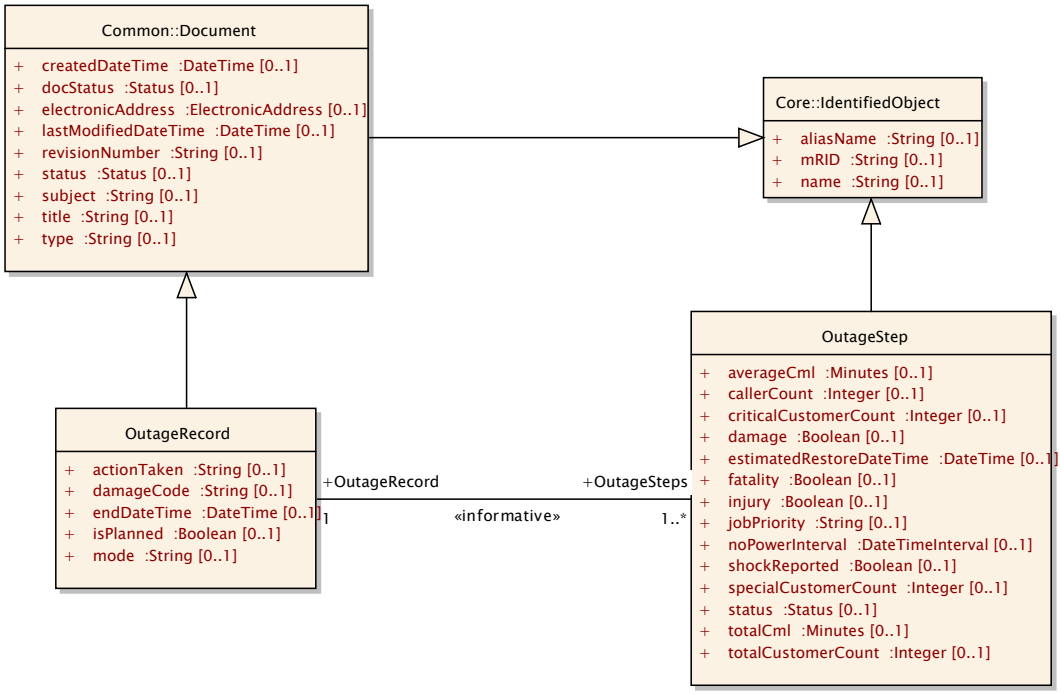

Figure 4.1: The inheritance hierarchy of the *OutageRecord* class, and it's association with an *OutageStep*.

domain. When creating a new implementation, however, the allowed content of *NameType.name*, which identifies the *NameType*, is not defined. Later in this thesis the *NameType* is used to link a SCADA code to an object. However, the standard does not define an unambiguos way to do this: on one system, the *NameType.name* of a SCADA code could be 'SCADA_name', on another 'OPC_code' and on a third one 'etaohjaus_koodi' and so on. Due to this, additional coordination is needed between vendors to create interoperable implementations. The naming class relations are shown in figure 4.2.

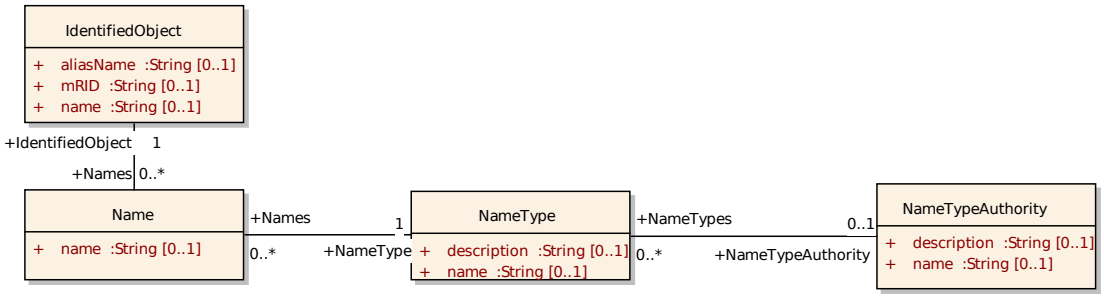

Figure 4.2: The naming properties in the CIM model. Graphic is exported directly from the IEC 61970 15v31 draft UML model.

4.1.2 Standard development

While the IEC standards themselves are human-readable documents, an integral part of the CIM standards is an Enterprise Architect[1] UML file containing the data definitions. Large portions of many of the CIM standards are duplicating content present in these EA models, and in the actual task of implementing an interface these files are very important. These files are available even as work-in-progress revisions for CIMUG members.

A CIM profile is a collection of rules as to which CIM objects should be used in a certain context and which properties are required. Technically, a profile is often presented as an XML, RDF or OWL schema file, but a human-readable document containing these rules is also a profile in a broad sense.[2] An IEC standard giving guidelines as to the usage of CIM is a profile - and indeed a schema file is appended to many parts of the CIM series of standards. In addition to these, specific profiles may be created for use in individual systems. Even as profiles outline different requirements for data, the base data format is still shared between all profiles, so the aim of a standard interface is not lost.

The CIM model as distributed by the CIM UG is as of 2011 still ongoing a lot of changes and frequently breaking compatibility with older revisions. IEC-published standards set the model at a specific format, but at the current rate of development basing features on a published standard will cause a huge maintenance overhead when the standards are updated to the latest model versions. Therefore most work on this thesis will be based on a recent Enterprise Architect file release and not on the already outdated standards. When referring to a specific CIM revision, the convention is to use the major revision number of the IEC 61970 model. See figure 4.3 for a reference on how the version number is currently expressed in the CIMUG EA model file names.

Recent major changes include a new naming system, changes to the transformer model, a more detailed phase-wise modelling capability and many other additions to the model. A lot of work has gone into converging network model formats of distribution and transmission into a single base format. A detailed change log is distributed by the CIM UG with every model release.

It is hard to see a standardized CIM gaining widespread adoption as long as even fundamental parts of the model see significant, compatibility-breaking changes on an annual basis. Regardless of this, even the current non-stable model works well as a basis of building application-specific implementations, as vendors can simply

[1] EA is a proprietary software for creating Unified Modeling Language diagrams and associated data models. The software also includes tools for code generation.

[2] However, according to the standard, a profile "may be expressed in XSD, RDF, and/or OWL files"

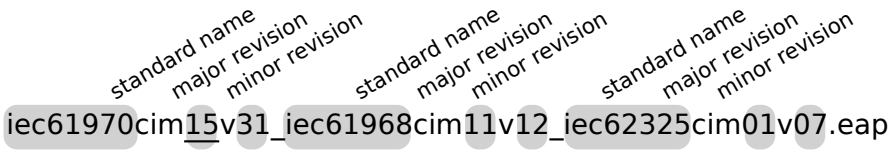

Figure 4.3: The CIM version is usually referred to by a number, which is the major revision number of the IEC 61970 data model. This figure shows a typical file name for a CIMUG EA model. Note that typically all the standards of the CIM family share a single EA model, as the data types contain cross-references.

agree to base a new interface on a specific CIM model release.

4.1.3 CIM interfaces

In the current state, CIM only provides a data model, but no specific interfaces. Some parts of the standards, such as IEC-61968-9[3] provide a profile that lays out the parts of the CIM model to be used in defining interfaces, and even some non-normative directions for the interfaces in the form of WSDL schemas. Even this is still not enough to make a plug-and-play interface just by following the standard.

It has been proposed that the CIM standards should not even try to go as far. Instead, the standardization effort should focus on perfecting the data models, while implementors would freeze a subset of the model to be used in the implementation as a profile. Further, implementors could take advantage of XML constructs such as namespaces to allow software to know which subset of the CIM is in use. [13]

4.2 Other data exchange standards

This section lists some standards specifically applicable to the operating environment of DSO:s. With the exception of MultiSpeak, these are not direct alternatives for the CIM family, although some overlap does exist. In addition to standards listed here, many more exist for general business purposes. Beyond standards, organisations exist that publish guidelines for selecting and implementing them, a finnish example being TieKe[4], a non-profit organisation for the advancement of computing technologies in the society.

4.2.1 Geographic data exchange standards

The network management systems like a DMS and AM are actually lightweight GIS packages, and the term 'lightweigh' is becoming increasingly less applicable. Therefore easy geographic data exchange and utilization is becoming more and more important.

[3]Interfaces for meter reading and control
[4]Tietoyhteiskunnan Kehittämiskeskus [14]

Simple positioning of components on a map does not require elaborate service based map exchange. For simple operations, manually importing map data periodically suffices. Recently the use of governmental land use guidelines as basis for outage worth calculations has been suggested in Finland. [15] The same data may in future be required for investment calculations as well. As this kind of data may be frequently updated and failing to use up-to-date data may result in a breach of legal requirements, automated map data updates may become a customer requirement for DMS vendors.

Organizations providing geographic data exchange standards include the Open Geospatial Consortium and Open Mobile Alliance. A source of geographic data, the OpenStreetMap also defines their own data structure.

The Open Geospatial Consortium, is a standards body formed by over 400 companies – including names such as Google and Oracle. Their publications include multiple standards for exchanging geographic information, among them GML and WMS.[16]

GML, or Geography Markup Language is an XML-based format for defining data in a Geographical Information System (GIS). The standard covers 2- and 3-dimensional data, time-dependent properties, different map projections and complex shapes. The GML does not specify a service interface. [17] Previous versions of the CIM base model referred to GML datatypes on representing geographical data.

WMS, or Web Map Service, on the other hand aims to define a standard HTTP[5] interface for fetching map data from a service or multiple services. [18] Currently the DMS600 map handling is implemented internally, with importers of multiple map types. In most current DMS600 installations map data is held on a central server from where it is cached to workstations manually. Applying a protocol such as WMS would decouple part of the map maintenance from the DMS600 suite and simplify data sharing over multiple systems.

The Open Mobile Alliance Mobile Location Protocol defines an XML format for transferring location data of mobile devices. This standard could provide usable data especially for work group positioning features in a DMS. The MLP does not define a transport protocol. Because of the nature of XML, data formatted in MLP would be easily translatable to CIM data using simple XSLT, or equivalent, transformations. The CIM data objects that might be constructed from MLP data would be especially *Crew* objects, which reference the *Location* object. The *Crew* object is likely to be defined in the upcoming part 6 (Interfaces for maintenance and construction) of IEC 61968.

OpenStreetMap[6] is a service aiming to create free global streetmaps that are

[5] Hypertext Transfer Protocol

[6] `http://www.opensteetmap.org`

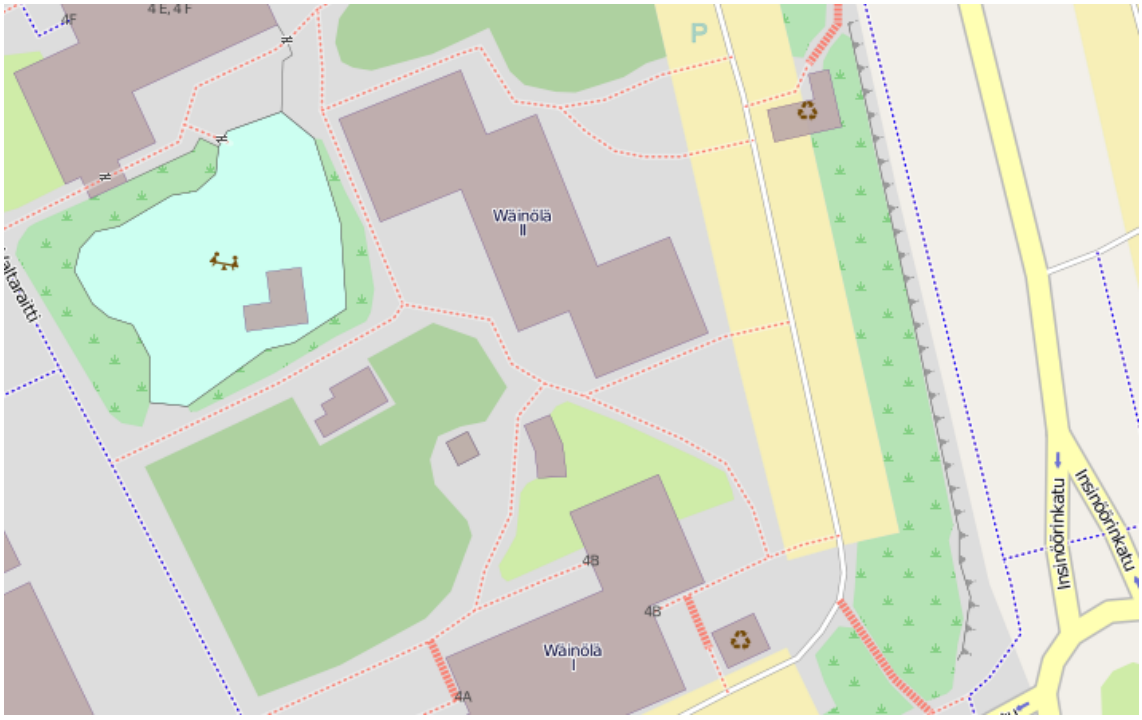

Figure 4.4: An example map from OpenStreetMap, showing an area with above-average detail.

uncumbered by copyrights. Due to the open nature of the project, the quality of the map data varies greatly with location, being at best very detailed as shown in figure 4.4. The service provides a proprietary XML format export of their data. Using this data as maps in a DMS/AM system might prove an interesting case study. The project has at least some coverage for locations around the world, so a demonstration installation capable of reading OSM data would be able to legally access local maps free of cost. In a production deployment the costs acquiring proprietary map data and implementing the required conversion routines is a lesser part of the expenses involved in deploying a DMS/AM system and therefore using such free data is less intriguing. The OSM provides a very simple XSLT stylesheet for transformation to GML, so implementing a GML importer would allow usage of this free data.

4.2.2 MultiSpeak

Developed before the CIM model by National Rural Electric Cooperative Association, MultiSpeak serves a very similar if not as broad purpose as CIM. The first MultiSpeak specification was released in late 2000. While the CIM model is designed to be used in transmission, generation and distribution operations, MultiSpeak mainly focuses on distribution. [19]

Another key difference is that MultiSpeak lays out the communication protocols and markup used in a normative fashion. The transport layer defined by MultiSpeak

is SOAP over HTTP, or alternatively a file-based transfer. XML is used as the markup language.[20, p. C-1] Where an XML markup is used for implementing the CIM model, a direct translation between CIM and MultiSpeak can be achieved using stylesheets and readily available tools. This does not guarantee completely compatible semantics, but an IEC working group is active in producing mapping guidelines between the two. [21]

Multispeak defines three communication types: batch, request/response and publish/subscribe. Encryption of communications is not required by MultiSpeak, but SSL security is allowed where desirable.[20]

Being more normative in it's choice of supported technologies, MultiSpeak has matured into a standard directly usable for new implementations. A central authority performs compliance testing of products and a number of reports have been published. According to NRECA, MultiSpeak implementations are used at over 250 utilities as of early 2011.

Although international recognition is claimed by NRECA, MultiSpeak utility members are exclusively US companies as of 2011. [19]

4.2.3 IEC 61850

The IEC 61850 series of standards defines functionality for substation automation. It partly overlaps with the CIM standards, and as such a mapping between the two is in progress.

The IEC 61850 standards differ from the CIM in that they provide more refined solutions to smaller scale problems through strict semantic rules, whereas the CIM mostly defines data types that can be utilized more loosely. Certification is available for IEC 61850 compatibility for vendors. The aim is to provide a more plug-and-play compatibility between products.

MicroSCADA, the SCADA system of which the DMS 600 is part of, contains some support for IEC 61850.

5. PREVIOUS IMPLEMENTATIONS OF THE CIM STANDARDS

Due to the broad nature of the CIM standards, a single application is unlikely to include the whole CIM model is practical use. This chapter lists some known existing deployments.

5.1 In MicroSCADA pro DMS 600

During the past few years, some features to support interoperability through the CIM standards have been developed in the DMS 600. Most notable is a network data export to CIM/RDF format, done as a masters thesis. This work has partially steered the development of internal data models to be more compatible with the CIM model, although they remain proprietary and direct mapping or a move to use the CIM model internally is not a goal. [22]

In addition to the MicroSCADA pro DMS 600, another ABB product, ABB Network Manager, utilizes the CIM model for data exchange.

5.1.1 An AMI interface to DMS600

A new AMR interface was created concurrently with this thesis, and the design was a point-to-point SOAP protocol with a subset of the IEC 61968-9 data types being utilized. In the future it is expected that this interface can be converted into a real SOA deployment with trivial development work. The author was involved in the specification work, while the instructor was representing ABB in the project. The interface supported relaying alarms and remote connect/disconnect instructions between the DMS and the AMI. Billing and metering functions were implemented in a separate project.

The interface was specifically developed for a single distribution network operator. In addition to representatives from the utility, the project spanned an AMI vendor and a 3rd party system integration consultant. Actual implementation of the DMS side of the SOAP interface was contracted to 3rd party developer.

The architechture is explained in figure 5.1. While the figure shows a message bus and an existing Microsoft BizTalk installation was leveraged, the actual role of the bus for this implementation is to just act as a message queue on a point-to-point

link. The CIM model is partly used in the SOAP interface between the DMS600 web service interface. The bus is administered by a third-party integration consultant. A proprietary protocol is used to interface the AMI system and the bus. The DMS 600 web service interface pushes the data on the proprietary DMSSocket protocol, which propagates them to DMS 600 WS instances.

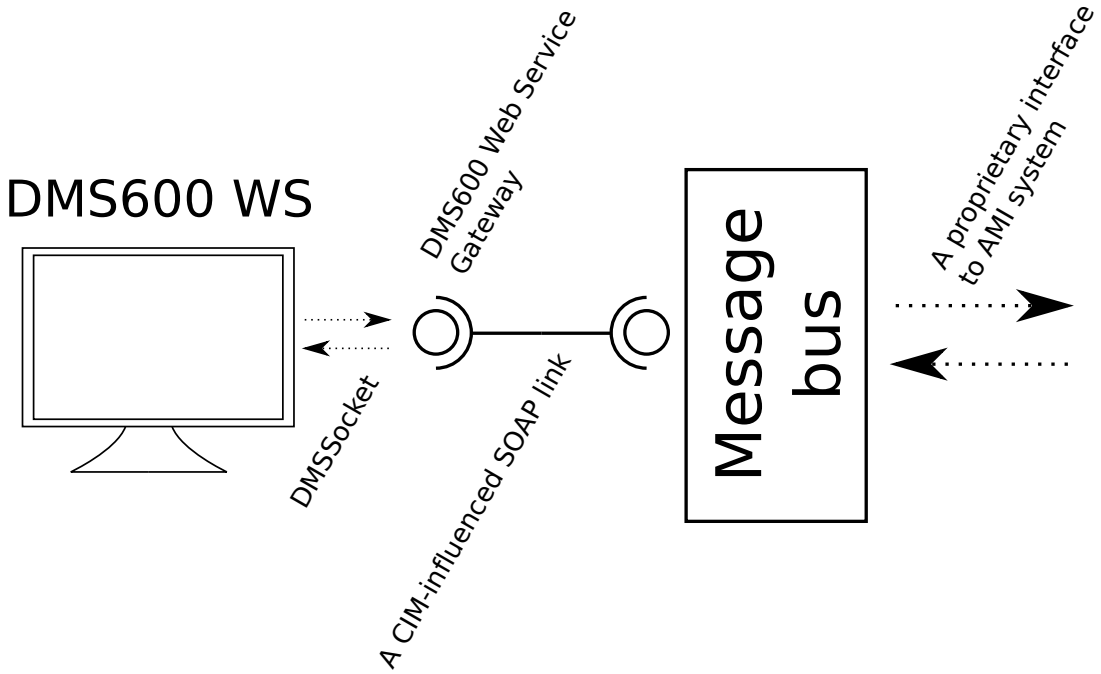

Figure 5.1: The AMR deployment where a CIM point-to-point SOAP interface was developed.

The point-to-point SOAP link is of interest to this thesis. The specification chose to support a few data types from the IEC 61968-9[1], most importantly *MeterReadings*, *EndDeviceEvent(s)* and *EndDeviceControls*. Only select fields of these object types were allowed, most important being the *category* field of *EndDeviceEvents*. For this field, a set of allowed values were defined based on annex E[2] of the standard.

In accordance with the schemas defined in the informative annex H[3] of the standard, plain XML was used for markup. The data structures required in this scope do not have cross references or other features that would call for RDF or OWL. Acceptable values for alarm types were specifically listed.

Where classes inherited properties from others, XML subtags were used. No reference, such as the RDF dot notation, was made to the parent class in accordance with plain XML/XSD convention. Code listing 2 shows a simple *EndDeviceEvent*.

[1]Application integration at electric utilities - System interfaces for distribution management - Part 9: Interfaces for meter reading and control

[2]Recommended EndDeviceEvent category enumerations

[3]XML Schemas for message payloads

In the CIM class model e.g. the property *category* is inherited from *ActivityRecord*, but in here it is not implied.

Code listing 2 An EndDeviceEvent in XML notation as used in the SOAP AMR interface. Schema references have been omitted for brevity.

```
<EndDeviceEvent>
    <category>6.25.1.160.128</category>
    <createdDateTime>18-01-2011 16:38:42+02</createdDateTime>
    <Assets>
      <mRID>METER_00</mRID>
    </Assets>
</EndDeviceEvent>
```

Listing 2 also shows another notational decision made. While the annex E[2] defines a system of creating values for the *EndDeviceEvent.category* field, a way to signify a fault indicates only a specific phase is expicitly defined. In the end, phase notation – 128 signifying phase A – from anned C was used as an appendix to the category code.

As both endpoints of the SOAP pipe were developed in co-operation, it remains to be seen how trivial it would be to make the interface compatible with a separate AMR implementation built on top of the IEC-61968-9 standard. The standard does define the syntax of the messages clearly, and therefore these would not be expected to cause incompatibilites. The standard gives, however, very scarce instructions on the semantics of the messages. Therefore it is expected that the content of messages from a separate implementation would not be understood or even misinterpreted. This was understood throughout the project, and although supporting all the possible messages was not even attempted, the implementation was made such that extending it is trivial.

In the end, the specification for the interface turned out to be comparable to size to what a from-scratch implementation would have been. Embracing CIM is expected to pay out later, as now the interface at least is similar to others based on the CIM. In this case, the same interface will be utilized by many separate AMI systems. Being based on a standard, it has been easier to convince other vendors to adopt this specification.

5.2 Other vendors

Other vendors are increasingly adopting the CIM standards. This section lists some products that claim support for CIM. In addition to these, many small applications, such as CIMTool[23], CIMSpy[24] and CIM EA[25] exist. These tools are helpful in validating and viewing models.

The Siemens PSS is a family of software packages for network operation and planning. Among the components are PSS/E, an application for planning and operation of transmission networks, and PSS/SINCAL, a calculation tool for electricity and pipe networks. The Siemens PSS applications include CIM network model import and export functionality, as well as tools to manage the CIM models.

DIgSILENT[26] makes software for power system simulation. CIM is among their supported data exchange standards, although currently only transmission network profile support is claimed. DIgSILENT products aim to connect to SCADA, GIS and other applications through an Enterprise Service Bus and provide advanced simulation features to supplement existing software.

5.3 Network operators

Some network operators have conducted research in cooperation with manufacturers to allow more interoperability in their systems. This section lists some projects that have been described in published articles. In addition to these companies, members of the CIM User Group include many network operators, some of which also operate a distribution network.

Companies operating transmission networks are generally much larger than the ones dealing with distribution networks. This has also made them active participants in the standardization effort.

One example of an active CIM user is Long Island Power Authority, a governmental non-profit owner of a transmission and distribution network in the state of New York, United States. The company serves 1.1 million customers, making them one of the biggest electricity network owners in the US.

The company has selected a business model in which they act as the asset owners in the electricity network, but outsouce all operations.[27] To gain the full benefits of outsourcing, the company needs to be able to change service providers with minimal overhead. As an asset owner, the company also retains ownership of all operational data. To avoid having to specify a software stack for their contractors, they have instead pushed through a project of creating a company-wide specification for data storage and transfer.

This project aimed to build a semantic model for all relevant data. The model was based on CIM, but apparently exceeded in scope the currently published set of CIM standards. In addition to CIM, the company reports to have used ANSI and ISO standards and MultiSpeak in creating their own data vocabulary.

The infrastructure at LIPA was selected to be a set of two separate, although bridged, ESBs. One was carrying the near-realtime data of the control center environment, the other used by asset management systems among other things and coupled with more advanced data warehousing.

The company concluded that their approach of creating a centrally managed data model was needed to clarify the many possible interpretations of the CIM and other standards. Also observed were 'significant' performance issues on their implementation where large models were involved. Data translations (propably XSLT) were identified as a main cause for suboptimal performance.

The project continues and the company expects to increase their flexibility in changing service providers and systems as needed. The project was estimated to have lowered future life cycle costs of their core operations. [28]

6. IMPLEMENTATION CONSIDERATIONS

Security considerations in a critical environment such as the control of the distribution network should be of similar level as is the security of the physical assets. When more and more different applications are implemented in the system, the security model of completely isolating the installation becomes more difficult.

The security considerations should prepare for both malicious and erroneous behaviour from both the users and the software. Malicious behaviour should not be singled out to only arise from actors outside the system. A person with some access clearance may attempt to cause behaviour they are not authorized for, and this has to be taken into account.

A thorough analysis of security in smart grid deployments was conducted by the United States National Institute of Standards and Technology in their report *Guidelines for Smart Grid Cyber Security*. This report divided the security threats in three domains: availability, integrity and confidentiality. [29]

Availability means the system must be accessible at all times, and operations must complete in a specific time. Though primarily a usability issue, availability becomes a security issue when timely access to data is required to maintain safe operations of a system.

Integrity means the origin of data must be known, data must not be modifiable by actors not authorized to do so and the data is known to be correct. The NISTIR[29] also specifically identifies time stamp data as something that should be confirmed to be correct.

Confidentiality means data in the system must not be accessible without authorization. This has both legal and business implications. There may be legal consequences for releasing eg. customer data. On the other hand, market data may cause harm to business pursuits.

6.1 Encryption and signing

Though other so-called symmetric methods are usefull in some situations, the basis of encrypting messages and verifying their origin in modern computer systems is based on public/private key, or asymmetric systems. The basic functionalities provided by the PPK[1] model are

[1] Public/Private Key

- Given a public key, a sender can encrypt the message so that only recipients possessing the corresponding private key can read it.
- Given a private key, a sender can sign a message so that all recipients who posses the corresponding public key can verify the integrity of the message.
- A public key cannot be used to obtain a private key.

These capabilities lead to further security considerations

- Private keys must not be compromised.
- Some method must be in place to verify the origin of a public key.

In a wide deployment such as the internet, the latter is often addressed using trust chains, where a public key is signed by the private key of a third party that can verify its origin, and the public key of said third party is then signed by the private key of another party that can verify the identity of the third party, and so on. In the end a recipient of the message should be able to find a link in the chain they can trust. In an internal deployment, such as a typical SOA at a DSO, the keys can be individually verified. More complex systems may be needed as the deployments expand across companies.

In a SOA deployment where encryption and signing are desired, one consideration is whether messages should be signed by a key belonging to a system (the software program), or a person (the user). Using a single key for each system probably results in simpler maintenance, but does not provide the same level of security. Additional security could be achieved by maintaining personal keys in a system such as a smart card deployment, which would ensure no private key ever resides in a computer.

When a message bus is deployed, some functionality of the bus may depend on its ability to read and process the messages that flow through. Allowing this weakens the security of the deployment, as illustrated in figures 6.1 and 6.2. Note that the same consideration also applies for message encryption, even though the captions refer to signatures.

6.2 Denial of Service

Denial of Service is a type of attack where the availability of information or a control system is compromised by an attacker blocking access to legitimate users. A similar type of failure may be caused by malfunctioning software, such as a subsystem creating too many requests in a short time.

A denial of service attack may happen either by sending a massive amount of legitimate requests, or by triggering a bug in the software and therefore causing it to enter an infinite or longer-than-expected loop in processing the request. The

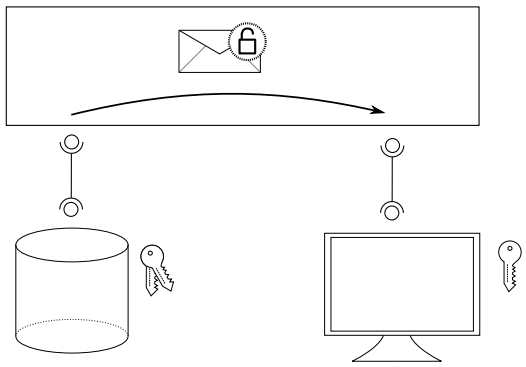

Figure 6.1: Signing messages in the bus. In this example, the sender system signs a message using their own private key. The bus is unable to make modifications to the message, so intermediate processing is impossible. The recipient knows for certain which system sent the message.

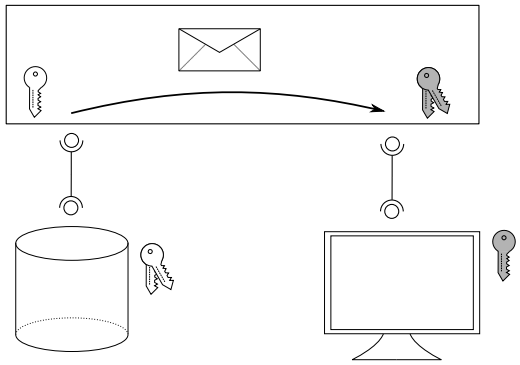

Figure 6.2: Here the sending system signs the message as before, and the bus verifies the signature to make sure the first transmission link was uncompromised. But in this example, modifications are made to the message in the bus, thus invalidating the original signature. The receiving system sees a message signed using the bus key, and is therefore unable to verify the real origin of the message. This creates a single point of failure for the security of the system, the message bus.

Type of communication	Acceptable delay
Protective relaying	≤ 4 ms
Transmission situational awareness monitoring	< 1 s
Substation and feeder SCADA data	seconds
Monitoring of noncritical equipment, some market pricing information	minutes
Meter reading, longer-term market pricing information	hours
Collecting long-term data such as power quality information	days/weeks/months

Table 6.1: Acceptable time delays in a smart grid environment from a security standpoint. [29]

former can be prevented by setting limits on the number of requests allowed by a specific subsystem, but the latter requires validation and timely upgrades in case a bug is found.

The NISTIR outlines acceptable time delays for different types of messages in the system. These requirements are outlined in table 6.2. Using the limits in the table as guidelines, it can be conluded that some interfaces require advanced protection against DoS attacks, while for others it may be acceptable to assume human interference in case an attack is detected.

The IEC 61850-3, which specifies general requirements for the interfaces in substation automation specifies DoS attacks as a threat to be considered. A lot of the use cases for the CIM standards on the other hand fall into categories where time-critical operation is not required, but the threat should be considered on a case-by-case basis when implementing data exchange interfaces.

6.3 Data integrity

To achieve reliable operation of systems participating in a SOA deployment, it is important to ensure messages are not lost and that a single subsystem sends only messages that do not conflict with others. There exist WS-* extensions and message queue implementations that define features for ensuring messages are never left undelivered or pending for a long time without notifying relevant systems of the failure, and that a single message is not delivered many times.

Another part in ensuring integrity is validating the data being transferred. Syntactic validation means checking a message against a predefined set of rules, a schema. This is generally done automatically for basic message formats such as XML/XSD when a generic toolkit is used. Less generally adopted formats such as RDF and OWL also define schema formats that can be used to check syntax, but tool availability is more of a concern. Such document may pass XML validation,

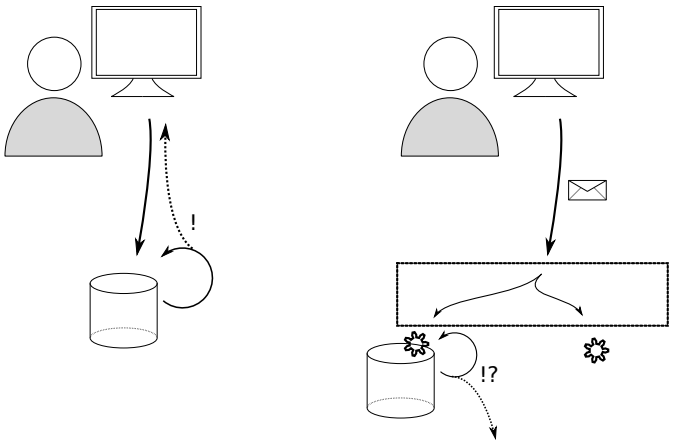

Figure 6.3: The problem of ensuring data quality in a SOA environment. On the left, a user inputs incorrect data on a traditional centralized system. The error is noticed in the validation procedures and the user is notified. On the right, a badly designed SOA environment propagates the command to two separate services that perform their own operations on the data. The other system detects the error, but no mechanism is in place to cancel the processing on the other service and notify the user.

thus being valid XML, but fail to adhere to specific requirements of the RDF or OWL schema.

Semantic validation refers to checking that the data itself is not only in the correct structure, but also correct in the specific sense of the domain, in this case power grids. Existing schema validation tools cannot be used to provide a complete set of validation rules for all cases where an engineer instantly sees a declaration cannot be correct. In current systems, a lot of data validation takes place on client side code. The DMS for example does not allow the user to draw a network that is not consistent. This issue should be considered when distributing parts of the system to separate services that do not see all of the relevant data at once to ensure no regressions happen in the feature set of the tools.

An illustration of the problem can be found on figure 6.3. The situation can be resolved by implementing a transaction system where the service endpoints receive and validate the message, but only proceed after all systems have notified the message can be processed and a commit order is issued.

6.3.1 Model Resource Identifiers

The mRID field in the CIM objects gives a globally[2] unique identifier for each object. A datatype commonly used for such identifiers is a 128 bit number, often called unique identifier in database management systems and visually represented as a string of 32 dash-delimited hexadecimal digits. The CIM does not mandate the use

[2]across systems

of this datatype, but it would be desirable to ease interoperability with databases.

The DMS600 database does not use the unique identifier data type for all identifiers, insted some use running numbers and others arbitrary strings. Objects of different types may have colliding identifiers in the DMS600 system, which is prohibited by the CIM. To overcome this, lookup tables will be needed to give each DMS object an identifier that can be guaranteed to be unique. In the long term, migration towards using unique identifiers internally is expected, not only for CIM compatibility but also for database performance and code maintenance reasons.

A related issue arises when synchronizing data between multiple systems, which may each have a separate internal representation of a single real-world object. This is undesirable, as conflicts in the data are imminent in a complicated system.

To mitigate this, a single 'authority' system can be selected for each datatype, the other systems being obliged to keep their data synchronized with this master system by listening to the relevant event broadcasts. To see an illustration of this problem, see figure 6.4.

On actual implementation a central repository should be created to allow lookups of the authority of each message type. Technologies such as UDDI are possible choices for this function, being augmented by an ESB suite to perform the routing. Part 454 of IEC 61970, which is currently in draft form, provides additional guidelines for creating such a repository.

6.3.2 Version control of network assets

As the CIM communication bus in theory relays all messages related to the operation of the distribution network, it seems an optimal place to implement a version control system of the network model and related documents. This functionality may or may not be an integral part of the communication system. In theory a separate program, that subscribes to related messages could implement this functionality. Many functions of the DMS would benefit from such a repository. More detailed historical operations records could be constructed from such data. New network designs could be more easily compared with installations already decommissioned.

Some version control is also built into the CIM objects themselves. A *Document* object for example has attributes *status*, *revisionNumber* and *lastModifiedDateTime*. Nonetheless, this is not sufficient to track previous version of the object.

Some libraries exist to create differentials of XML files, such as XML Diff by Microsoft[30], an open source solution diffxml[31] or a Java library by Oracle[32]. Unfortunately, all of these produce output in non-standard and incompatible formats[3]. Before an authority such as W3C comes up with a standard format, the

[3]with the limited exception of Oracles tool, which can output an XSLT file to transform from one version of a file to another

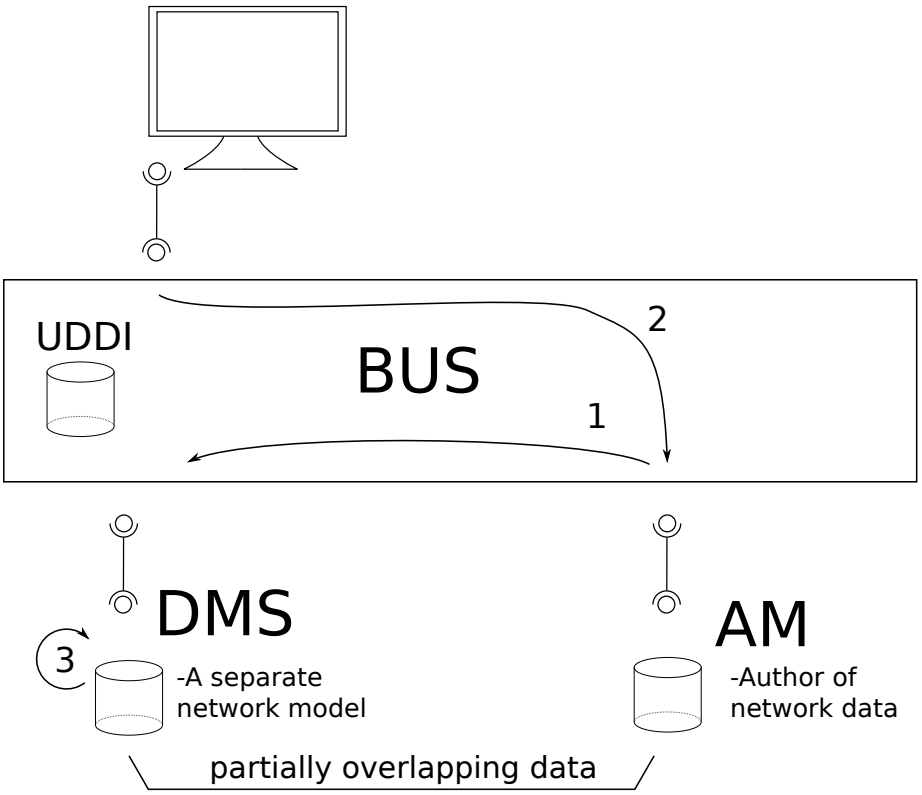

Figure 6.4: A clarification of the Object Author problem. In this example, the network model is held in two different databases, one for asset management purposes, the other for a distribution management system. New content is created on the AM database. Update notifications are propagated to the DMS database (1). Queries from separate systems are routed in the message bus to the correct service interface using a service discovery database (2). The problem arises when existing code is unaware of the service architechture and makes a modification to the data internally in the DMS database (3). This problem can be partially solved by inserting hooks to the relevant events in the DMS system to propagate change requests to the AM system. However, the AM system may in turn refuse the modification, but the existing codebase does not know how to handle the situation.

utility of a version control system based on XML differentials might be limited to internal uses within a one-vendor system.

An incremental model has been proposed specifically for CIM data by Langdale Consultants. It is implemented in some tools, such as CIMSpy. While the proposed difference model format itself is not specific to the CIM model, but is rather a generic RDF differential format, the proposal includes many useful conventions specific to CIM, such as instructions on naming conventions. [33]

6.4 Performance

The performance of a system is comprised of two components: the latency and throughput (or bandwidth). Latency means the maximum time a single message travels from a sender to the recipient. Throughput is the maximum amount of simultaneous messages the system is able to handle without a degradation in latency.

The acceptable latency from a security point-of-view was outlined in section 6.2, and these should be taken as the worst-case acceptance limits. From a usability viewpoint, the acceptable delays are shorter.

Estimating the throughput requirement can be done by estimating a worst-case scenario of simultaneous messages being transmitted in the system. One possible starting point is a large-scale outage that causes a majority of remotely read meters to trigger an alarm at once. If a major event would destabilize the whole power system, a transmission volume of

$$N_m s \tag{6.1}$$

can be foreseen, where N_m is number of messages and s is the size of a data package. Such an event would also cascade to other systems, creating a large amount of additional messages. In an event such as this, additional latency or even delivery failures may be acceptable, depending of the overal design of the system.

For normal operation, the required average throughput for a given system can similarly be obtained from

$$\sum_{i=0}^{N_m} N_{s,i} s_i f_i \tag{6.2}$$

where N_m is the amount of different messages transmitted, $N_{s,i}$ is the amount of senders for a given message type, s_i is the size of a given message and f_i is the frequency of a given message. The result of this calculation will be slightly optimistic, as it assumes all message types are transmitted at a constant interval. In reality transmissions would overlap. Estimating the effect of this is difficult without a real-life pilot deployment.

The throughput capability of a messaging system also cannot be analyzed only from a network transmission viewpoint. If any message processing or verification

is done on the ESB, the performance of these will likely cause a bottleneck instead of the network bandwidth. Performance problems in the processing steps may also be more difficult to solve by simple hardware upgrades, unlike transfer rate issues. The maximum allowable time to process a message to allow worst-case performance could be estimated by

$$\frac{t_l t_w}{N_m} \tag{6.3}$$

where t_l is the allowable latency for a given message, t_w is the time window during which the broadcasts will occur and N_m is the amount of messages being sent.

While rough, these considerations should be taken during the development of a system to ensure the development does not reach a dead-end in terms of achievable performance.

6.5 Available technologies

CIM leaves many implementation details open. While recommending XML/RDF on some points, details such as communication protocols are left for implementers to decide. Whether future revisions will enforce specific standards remains an open question. This chapter looks at some of the technologies - both standards and implementations - commonly used in realizing a SOA.

A central concept of SOA is platform independency. Although protocols are open, some pitfalls arise when using third-party libraries to implement them. Different libraries, or toolkits, support a different subset of WS-*[4] specifications, so a protocol implementation using a specific toolkit may depend on features not readily available on others. This effectively makes the implementation platform and API specific, barring a from-scratch implementation of the underlying WS-* protocol in use. Some consideration should therefore given to selecting WS-* protocols that have wide acceptance.

If an ESB suite is used on the deployment, significant work may have to be put to creating rules, transformation, orchestrations and extensions to it. These are not always transferrable to another ESB. Again, some open standards exist, but care should be taken to select an ESB that supports those standards, or accept that migrating to another ESB is prohibitively expensive in terms of reimplementation work.

6.5.1 XML

The markup language selected dictates the breadth of syntactic validation that can be performed using standard methods, and tool availability. Since XML has gained

[4] see section 6.5.3

widespread adoption, the question currently is not whether to use or not, but rather which extensions are most appropriate.

In some border cases performance of XML may not be acceptable due to its human-readable design. In such cases, some binary formats exist which aim to provide some of the benefits of XML while sacrificing human-readability. In other narrow implementations more simple markup standards may be appropriate, such as JSON[5] in web development.

The eXtensible Markup Language is a W3C[6] standard. It is intended as a structured data definition language. It is designed to be easily parseable by a computer but to also remain human readable. The XML standard itself defines markup rules, but the content and semantics is up to the user to decide. Accompanying standards define further rules for more specific use cases. The additional standards of the XML family increase the usability of XML, but an implemenation does not need to support all of them. [34]

Being computer parseable and having multiple implementations for the core parts of the standard, XML can easily be used in an application even without deep knowledge of the standards. For an example of an XML file, see listing 3.

Code listing 3 A simple XML file, the file type (the namespace) being *http://tut.fi/keskiketuri*

```
<?xml version="1.0" encoding="utf-8"?>
<Example xmlns="http://tut.fi/keskiketuri">
 <Text>Abc</Text>
 <Number>123</Number>
 <Number>256</Number>
</Example>
```

XSD is the preferred file extension of **XML Schema**, which is an XML-based markup for defining the allowed elements in an XML file. This information is primarily used for validating the contents of an XML file, but can be utilized by code generation tools as well. For an example of an XSD file, see listing 4. [35]

The main benefit of using XML Schemas is that data validation can be provided with little additional work. Many tools in development environments and ESB suites use XML Schemas as the basis of their operation.

Another XML extension, **XPath** is a way of making queries within an XML file. For example, the XPath expression */IdentifiedObject[mRID='abc']/name* returns the *name*-property of all objects of type *IdentifiedObject* with the *mRID*-property set to 'abc'. [36]

[5] JavaScipt Object Notation

[6] World Wide Web consortium, the standard authority of internet technologies

Code listing 4 A simple XML Schema, specifying an XML based file format called *http://tut.fi/keskiketuri*. An example file conforming to this schema was shown in listing 3

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://tut.fi/keskiketuri"
    elementFormDefault="qualified"
    xmlns="http://tut.fi//keskiketuri"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="Example">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Text" type="xs:string"
                minOccurs="1" maxOccurs="1" />
    <xs:element name="Number" type="xs:int"
                minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

XSLT or **XSL**[7] **Transformations** is an XML based format to model a transformation from one XML schema to another. XSLT uses XPath to define the transformations. An XSL file is a set of instructions that transform data from one file format to another. [37] An example transformation is shown in listing 5.

Some graphical tools exist for creating XSL transformations, for example Stylus Studio XSLT Mapper[38]. An ESB software could use XSLT to map messages between incompatible web service interfaces. Microsoft BizTalk uses a proprietary, partly compatible Biztalk Mapper file format to achieve a similar purpose[39], although other Microsoft tools, such as the .NET libraries and the Visual Studio development environment provide good support for XSLT.

6.5.2 RDF and semantic markup

The Resource Description Framework[8] is a W3C standard for describing allowed relations of data objects. While the RDF model does not specify the use of XML to represent the relations, the most common variant of RDF is the RDF/XML format where RDF relations are defined in an XML file. [40] [41] Other markup styles for RDF exist, for example RDF Turtle syntax, which aims to be more human-readable than RDF/XML.

As is the case with bare XML, RDF/XML also builds upon two types of files:

[7] XML Stylesheet Language, a set of standards consisting of XSLT, XPath and XSL-FO.
[8] RDF

Code listing 5 An example of an XSL Transformation to map from the previously defined *http://tut.fi/keskiketuri* file format to another one. The code inside the *match* attributes is in fact XPath. The terms *<* and *>* are escape sequences to the characters < and >, which are reserved characters in XML. The transformed file is shown in listing 6.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:e="http://tut.fi/keskiketuri"
  xmlns="http://tut.fi/keskiketuri#modified"
>
 <xsl:output method="xml" indent="yes"/>

 <xsl:template match="/e:Example">
  <AllNumbers>
   <xsl:apply-templates select="e:Number" />
  </AllNumbers>
 </xsl:template>

 <xsl:template match="e:Number[. &gt; 150]">
  <BigNumber>
   <xsl:value-of select="."/>
  </BigNumber>
 </xsl:template>

 <xsl:template match="e:Number[. &lt;= 150]">
  <SmallNumber>
   <xsl:value-of select="."/>
  </SmallNumber>
 </xsl:template>
</xsl:stylesheet>
```

Code listing 6 The output of the example transformation when applied to the example file. Another XML Schema could be provided for this file format, but is not included in this thesis for brevity. Note that schema definitions are not required for the transform to function, although it it uses the namespaces to qualify elements. Multiple namespaces may coexist in a single file, as in the case of vendor specific extensions to a standard file.

```
<?xml version="1.0" encoding="utf-8"?>
<AllNumbers xmlns="http://tut.fi/keskiketuri#modified">
 <SmallNumber>123</SmallNumber>
 <BigNumber>256</BigNumber>
</AllNumbers>
```

the RDF Schema[9] and the RDF data file. The RDFS file defines allowable relations between objects and the RDF file defines the data content of the objects. To process an RDF file, a schema is not necessarily required. The schema only provides validation information, but it contains no data. [42]

The CIM standards recommend the use of an RDF/XML file for representing network topologies. An obvious advantage of this is that RDF/XML provides a standard way of making multiple references to other objects. In bare XML/XSD objects can be defined, but no data type exists for creating and *validating* a reference to another object without making multiple copies of said object. While XSD validated bare XML files can specify identifiers and make references to those, an RDF Schema specifies which types of objects a specific property can reference. Code listing 7 gives an example of relation definitions in an RDFS file.

To further explain the difference between RDFS and XSD, a sentence can be used. An XSD can define a constraint on a property as such:

> A *TopologicalNode* must include a property called *Terminal*, the data type of which must be a string.

RDFS can describe a further constraint:

> The string value of *TopologicalNode.Terminal* must equal to exactly one *MRID* property of an object of type *Terminal* defined elsewhere.

While XML and XSD are very common standards and parsers, validators and editors exist for many different programming environments, RDF on the other hand lacks such tools. Some toolkits are in development to support RDF Schema validation and RDF file generation, but these are not mature enough to provide a common solution for all problems involving RDF. This is a serious hinderance for leveraging RDF. In limited use cases, such as CIM topology representation, not all features of RDF need to be supported and therefore a partial RDF implementation – be it third party or self developed – may be acceptable.

The Web Ontology Language[10] is a further extension of RDF designed to further describe ontologies. Ontology in this context means the description of relationships of objects such that new facts can be deduced from existing information. OWL aims to enable computers to do these deductions. The effort is part of a future development towards a so called semantic web, which aims to provide the capability of combining information from distinct sources. [43]

As an example, lets say there is an OWL schema defining the domain of educational ranks. Additionally, we have data stating a person is marked the author of a

[9] abbr. RDFS
[10] OWL

Code listing 7 An extract of an RDFS file used in defining network topology in the CIM format. The first paragraph defines *TopologicalNode* as a subclass of *IdentifiedObject*. The second paragraph defines a property called *Terminal* for *TopologicalNode*. The content of a *TopologicalNode.Terminal* has to be an object of type *Terminal*, or a subclass thereof.

```
<rdf:Description rdf:about="#TopologicalNode">
  <rdfs:subClassOf rdf:resource="#IdentifiedObject"/>
  <cims:belongsToCategory rdf:resource="#Package_Topology"/>
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/
    UML#concrete"/>
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/
    UML#byreference"/>
  <rdfs:comment>For a detailed substation model a TopologicalNode
    is a set of connectivity nodes that, in the current network state,
    are connected together through any type of closed switches,
    including  jumpers. Topological nodes changes as the current
    network state changes (i.e., switches, breakers, etc. change
    state).
    For a planning model switch statuses are not used to form
    TopologicalNodes. Instead they are manually created or deleted in
    a model builder tool. TopologialNodes maintained this way are also
    called "busses".</rdfs:comment>
  <rdfs:label>TopologicalNode</rdfs:label>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/
    rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:about="#TopologicalNode.Terminal">
  <cims:stereotype rdf:resource="http://langdale.com.au/
    2005/UML#aggregateOf"/>
  <rdfs:comment>The terminals associated with the topological
     node.   This can be used as an alternative to the
      connectivity node path to terminal, thus making it
      unneccesary to model connedtivity nodes in some cases.
      Note that the if connectivity nodes are in the model,
      this association would proably not be used.</rdfs:comment>
  <rdfs:label>Terminal</rdfs:label>
  <cims:inverseRoleName rdf:resource="#Terminal.TopologicalNode"/>
  <rdfs:range rdf:resource="#Terminal"/>
  <rdfs:domain rdf:resource="#TopologicalNode"/>
  <cims:multiplicity rdf:resource="http://iec.ch/TC57/1999/
    rdf-schema-extensions-19990926#M:0..n"/>
  <rdf:type rdf:resource="http://www.w3.org/1999/02/
    22-rdf-syntax-ns#Property"/>
</rdf:Description>
```

masters thesis, the publication status of which is 'pending'. From this data and the OWL definition a generic computer program can deduce that the educational rank of this person must be 'bachelor of science'.

Just as RDF, OWL also does not mandate the use of XML for defining the ontologies, but an XML syntax is defined which extends on RDF/XML. The CIM User Group distributes some OWL schemas. As of date, the advantage they provide over the plain RDF Schemas is for example that they define cardinality constraints, such as *A ConductingEquipment always has exactly two terminals.*

Due to the way RDF and OWL link data, queries can be run on top of the data. The main difference compared to the established SQL[11] is that object relationships are already defined by RDF, whereas SQL queries must define such relations for every query using the JOIN construct.

For example, to estimate what percentage of outages could be cleared faster if additional remote-controlled switches were deployed, it might be useful to list all previous unforeseen outages which required operating a manual switch. A human-readable equivalent of this query would be

> Fetch all *OutageRecord*s having *isPlanned* 'false', which link to a *WorkTask* of *type* 'manual switch operation'.

SPARQL is one such query language specified by the W3C[44], which resembles SQL in syntax. Additional specifications exist for representing query results in XML format[45] and for passing queries over a Web Service[46].

6.5.3 Simple Object Access Protocol

SOAP[47] is a standard for defining messages for service transactions. A SOAP message consists of an envelope holding a separate header and body. The header includes metadata required by the service interface while the body contains the actual request, or data transfer. The SOAP standard is a W3C recommendation and enjoys wide adoption.

SOAP messages are formatted in XML and typically transmitted over HTTP, although other transports are possible. Additional standards and specifications extend the vocabulary carried in the SOAP headers. These standards are often referred to as Web Service technologies, or WS-*. The endpoints of a service exchange reference these specifications by XML namespace definitions. To enforce an additional specification in the message exchange, the endpoints can define a property *MustUnderstand*, which requires that the other party responds with a fault in case it does not support the given header tags. SOAP also defines ways of transferring error messages, or SOAP exceptions.

[11] Structured Query Language, a query language used in database management systems

Some widely adopted extensions to SOAP, or WS-* standards, are

WS-Addressing A W3C standard by Microsoft, IBM, SAP, Sun Microsystems and BEA, WS-Addressing[48] defines vocabulary that enables asynchronous replies to SOAP messages. It allows definition of endpoint addresses, usually SOAP function URL:s to which replies are to be delivered.

WS-ReliableMessaging WS-ReliableMessaging[49] is published under OASIS[50] and defines an extension to provide validation constraints to SOAP messages. It allows ensuring that a sequence of messages is received in correct order, that all messages are actually received exactly once and they are received within an allowable timespan. A prior OASIS standard, WS-Reliability[51] exists for similar purpose.

WS-Security Although SOAP messages can be transmitted over a channel secured at the transport level, WS-Security[52] adds extensions to allow signing and encrypting SOAP message exchanges at the content level. WS-Security is published through OASIS. WS-Security introduces a significant performance penalty over transport-level security. Another standard, WS-SecureMessaging allows for securing conversations of multiple messages to be secured with a single key exchange, increasing performance. [53] Other related standards include WS-Policy[54] for defining acceptable security models, WS-SecurityPolicy[55] to extend WS-Policy to define additional policies such as requiring transport-level security and WS-Trust[56] to allow for exchanging security tokens.

WS-Coordination WS-Coordination[57] defines vocabulary for distributing transactions across multiple services. Another standard, WS-AtomicTransaction[58] defines further message attributes for defining transctions, ie. exchanges that involve multiple service invocations that should only be committed in case all of them are successful.

WS-Enumeration WS-Enumeration[59] provides tools for transferring large datasets in chunks, similar to transferring rows of an SQL query.

6.5.4 Service Discovery and metadata

Service discovery refers to publishing details about services that are available in a given network at a given time. This allows clients requiring a function to only know how to search for a service, instead of knowing the exact details of each service it references. Service discovery may be used to identify which service to invoke in the case a client finds many that provide the same functionality. This feature may be

useful for example in mapping a certain function to a certain software system, even if many are available that implement it: e.g. multiple systems may provide a function to query certain data, but it may be that only one system is kept up-to-date in a given installation.

The main solution to defining service interface syntax is WSDL, or Web Service Definition Language. It is an XML markup for defining service protocols. A WSDL definition is a single file which includes a list of available functions and the message format definitions (parameter/return value) using XML Schema. Tools are available to generate client side code given a WSDL definition for multiple programming languages.

To exchange the definition, WS-MetadataExchange[60] provides vocabulary for requesting service descriptions, namely WSDL files from running services. Some frameworks use a simpler method of altering the service URL to allow for HTTP requests for downloading metadata while WS-MetadataExchange defines SOAP messages to do this and therefore increasing interoperability between web service toolkits.

Two main approaches exist for the problem of service discovery. UDDI, Universal Description Discovery and Integration[61] provides service discovery using a central repository. An UDDI repository implementation is included in many ESB suites, among them Microsoft BizTalk.

An alternative to UDDI, WS-Discovery is a distributed service discovery protocol. Instead of a central repository, services use broadcast messages to notify others of their appearance and disappearance.

6.5.5 Web Service frameworks

To ease the workload in implementing and maintaining a service, it is useful to leverage an existing library for handling the protocol-level communication. This section lists some such frameworks. Table 6.2 shows a feature matrix for these implementations. The data in this table is compiled from web pages of the vendors, so cannot be considered fully reliable.

Main considerations in selecting a WS-* framework are the feature support, ease of development and ease of extending implementations to support WS-* specifications not included in the standard feature set.

Apache The Apache Software Foundation provides two separate WS-* frameworks. Axis2[62] is an open-source toolkit from the Apache project, primarily known for their widely adopted web server. Development can be done in both Java and C languages. Tools are available for the Eclipse IDE. Another Apache offering, CXF, was created as a merger of XFire and Celtix[63] WS frameworks, Apache CXF[64]

provides a toolkit for Java and Javascript languages. The list of available transports is interesting and includes such protocols as Jabber, originally an instant messaging protocol.

gSOAP gSOAP is a WS framework for C/C++ distributed under an open-source license, but with optional proprietary licensing available from Genivia for commercial developers. The library is compatible with multiple operating systems, and according to their website is adopted by numerous large companies. [65]

Metro Metro[66] is a WS toolkit developed as part of the GlassFish ESB, which was originally released by Sun Microsystems and later moved to Oracle with their acquisition of the former. Development plugins are available for both Eclipse and NetBeans IDE. The stack can be used in Java applications. A part of the project is Web Service Interoperability Technologies (WSIT), which aims to provide better compatibility between Java and .NET implementations, namely WCF.

Windows Communication Foundation WCF[67] is the WS technology offered by Microsoft. As such, the development tools and libraries are tightly integrated to other Microsoft products such as Visual Studio. The libraries are distributed with modern versions of the Windows operating system. Development can be done in languages supported by the .NET platform, namely C#, C++ and VB.NET. A partial implementation is available as part of the open-source Mono project, which aims to provide binary and source level compatibility with .NET for multiple operating systems[68].

6.5.6 The communication bus

An ESB, or Enterprise Service Bus is a software suite that handles delivery of messages in a SOA environment. It is not a necessary part, but using one may easy deployment by providing easy to use configuration and logging tools.

Typically, an ESB contains message queues that handle some of the task of assuring message delivery and consistency. Another part of an ESB is an orchestration engine, which means a specific message can be configured to trigger a set of operations (e.g. service invocations) and perform transforms on the messages. In addition, the ESB suite may contain additional services, such as an UDDI implementation.

Communication with an ESB can be done using various methods, such as file transfer, proprietary APIs or WS-interfaces. An ESB is often designed to allow handling of data in legacy formats, so they support many types of protocols.

The choice of an ESB most likely is not one the system vendor does, as these systems may already be deployed on the environment for other purposes. If that is

	WS-Addressing	WS-ReliableMessaging	WS-Security	WS-SecurityPolicy	WS-SecureConversation	WS-Coordination	WS-AtomicTransaction	WS-Trust	WS-Policy	WS-MetadataExchange	WS-Discovery	WS-Enumeration
Apache Axis2	✓	✓	✓			✓	✓		✓	✓		
Apache CXF	✓	✓	✓	✓	✓					✓		
gSOAP	✓		✓								✓	✓
Metro	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
WCF	✓	✓	✓								✓	

Table 6.2: A list of some available Web Service frameworks. If a framework does not support a given WS-* specification, that support must be implemented on a case-by-case basis.

the case, the system must adapt to whatever platform is in place. When an ESB is selected, care should be paid to it supporting standard formats for things like orchestrations in order to not lock development work on a single platform.

Some notable ESB suites include Microsoft BizTalk, IBM WebSphere and TIBCO ActiveMatrix. Open source implementations exist, for example from the Apache Software Foundation.

7. IDENTIFICATION OF SERVICES IN DMS600

This section attempts to analyze the current architecture of the DMS 600 system and identify parts of functionality that would benefit from a migration to a more service-oriented architecture. While most of the use-cases here are either already implemented or possible to implement without a new architechtural concept, they are good candidates of considering whether Service Oriented Architecture is worthwhile.

7.1 The internal architecture of DMS 600

As described earlier, internally the DMS 600 system exchanges data between workstations by means of direct database connections and proprietary TCP/IP socket messages. External interfaces connect to either, but as a rule-of-thumb real-time data goes through the the socket protocol and periodic batch transfers access the database.

The DMSSocket protocol is the main synchronisation strategy in current DMS 600 releases. It propagates network state changes to all workstations running on a system and acts as an input channel for some data from external sources.

For external systems to receive information of state changes within the DMS 600, the DMSSocket messages would have to be translated to a common format and relayed to them. Due to the nature of the DMSSocket messages, simple methods such as XSLT cannot be used, instead a runtime has to be built to handle the transformation to each new protocol. This situation is illustrated in figure 7.1.

In future deployments, where more complicated data structures are needed, it should be considered if extending the DMSSocket protocol is wise, or if it should be gradually superceded by an XML-based transfer format, likely one based on CIM. The obvious drawback is more complicated internal protocol handling in the DMS 600 applications. This is illustared in figure 7.2.

A third option is possible: the internal data transfer continues to use the methods that are simple to implement in the applications. Shared database methods are used for more complicated data types and the DMSSocket protocol remains to be used for simple internal messages. A CIM/XML-based interface is created on top of these, and external clients always connect to the CIM interface. In this scenario, a CIM interface only needs to be created once. Additional XML-based interfaces can be implemented using XSLT, without additional code. Future extensions to

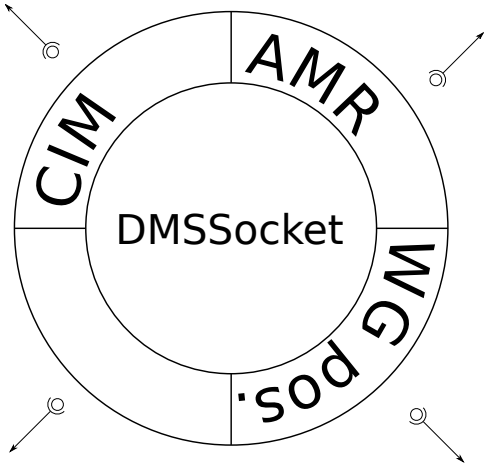

Figure 7.1: Current DMS 600 integration strategy. External messages are transformed to the internal DMSSocket protocol. The protocol is not XML-based, so common methods such as XSLT cannot be used.

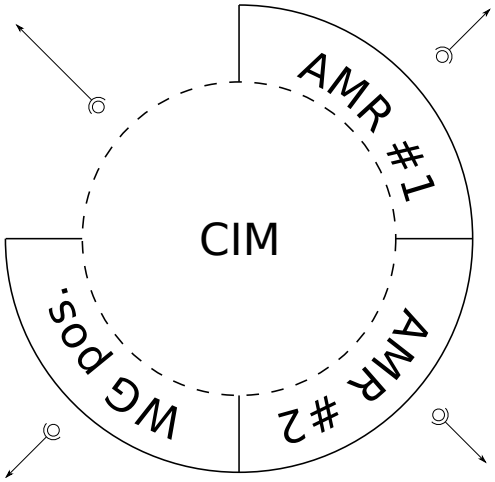

Figure 7.2: Possible future protocol layout. The internal protocol is XML-based, so XSLT can be used for mapping external XML interfaces without custom code. Direct interfaces to the internal protocol may also become possible if the CIM datatypes prove to be specific enough to allow interoperable clean-room implementations.

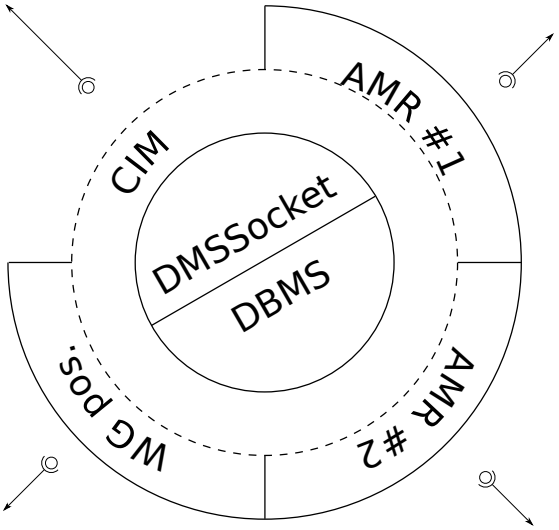

Figure 7.3: A synthesis of the possible implementation strategies. Internal messaging continues to use simple methods, but a CIM-based wrapper interface is built on top to allow leveraging XSLT for future interfaces.

internal protocols may choose to use the CIM layer directly, if it is perceived to offer advantages. This strategy is illustrated in figure 7.3.

While the third option seems most complicated, it would follow the principles of gradual improvement. Continuing to use the DMSSocket protocol as the external interface would be very laborous as the number of interfaces grows. A complete rewrite of the internal communication would be hard to justify as well, when the current implementation is fast and mature.

Currently client applications in the DMS 600 environment place direct database requests to the central server, or in a distributed DMS system to an area server. While this approach is fast, it somewhat limits the user account restrictions that can be enforced, as the user running the application has to have either direct or indirect access to the credentials required to run database commands. Restrictions like limiting access to specific tables can be done in the current system, but more detailed limits such as geographic user access limitations are impossible to enforce if the user is determined to connect using a separate program to directly access the database. In other words, some user access controls are implemented on the client side, which is inherently insecure. In some environments, this design is mitigated by running the clients in a controlled sandboxed environment, thus ensuring the access controls of the DMS client software cannot be bypassed.

Real benefits would be gained if the allowable database commands were wrapped in a service and the user only held the required credentials to communicate with this service, not the database. This approach would propably be slower than direct database queries, especially on operations involving large amounts of data to be transferred or those requiring small latency, such as operations involving a user interface. Yet, on systems where strict security is a requirement, sandboxing the application on a remote server creates even worse performance overhead and the possibility of doing data caching to avoid hitting the actual database could actually improve performance for those requests that are frequently done by multiple clients.

In addition to the security issues, a benefit of wrapping all database accesses would be stability of the client programs in the case of database connection failures. In a separate service it would be easy to retry connections in case the server is down, introducing some latency but possibly avoiding connection errors to be shown to the end-user. Codebase maintenance could also become easier as all calls to specific data types would exist in a single place, the wrapper service. On the other hand, the additional complexity could just as well make maintenance more difficult.

The current codebase for the DMS 600 Workstation alone contains over 700 calls to read data from the database. Rewriting all of this would be no small task, but could be done in incremental steps. In addition to rewriting the calls to instead contact a service, some user interface logic might have to be redesigned to allow for

greater latencies.

7.2 Possible functionality to be implemented as services

This section looks at future functionality that would be difficult or less efficient to implement using the current architecture. These proposals are mainly expansions of existing features. New interface requirements with external systems are not discussed here, as the merits of a SOA approach to common integration scenarios has been communicated in previous chapters.

7.2.1 Processing AMR events

Automated Meter Reading deployments provide information from the low voltage networks which when combined with a topological network model can be used to deduct information about state of the medium voltage network. Knowing the phasor group of the transformer that a low voltage AMR meter is supplied by, specific phase voltage inconsistencies and unbalances are known to signify a specific type of fault in the medium voltage network. In some cases this information can help identify a fault difficult to detect otherwise.

ABB has previously involved in developing a pilot system for managing such events. In the pilot system the main part of the deductions were performed on the lowest level, the smart meter. [69] This section proposes moving some of the deductions higher up on the system. The previous approach requires a capability of remotely upgrading configurations in the meters if network topology changes, although such changes are often rare or non-existant - for example distribution transformers within a single system most likely all have the same vector group.

A good implementation on a higher level would include a simple way of defining parameters for these algorithms, such as voltage limits that are interpreted as a specific type of fault. An optimal solution would not have hardcoded detection algorithms, but instead allow the user to define new alarm/topology combinations that trigger a higher priority alarm.

One way of detecting these faults is running the detection algorithms on the DMS client side computer. For systems with multiple workstations subscribing to AMR alarms, the detection would run multiple times. Another possible deployment would be to have a separate service subscribing to AMR alarms with only the required part of the network model in memory. This service would run heuristics on the incoming alarms and send new, higher priority alarms as necessary. The proposed design is illustrated as a sequence diagram in figure 7.4.

Implementing the functionality in a separate service would allow the management of the parameters of the algorithms in a single place. This type of implementation,

if done in accordance with standard message formats, would be easy to migrate to different environments, while a DMS 600 built-in functionality would be tied to that platform. Additionally, the new alarms would also be propagated to possible other systems subscribing to AMR alarms. A DMS internal functionality would require yet another set of interfaces to be created to notify each system of such findings, or every system requiring AMR alarms would have to duplicate the functionality.

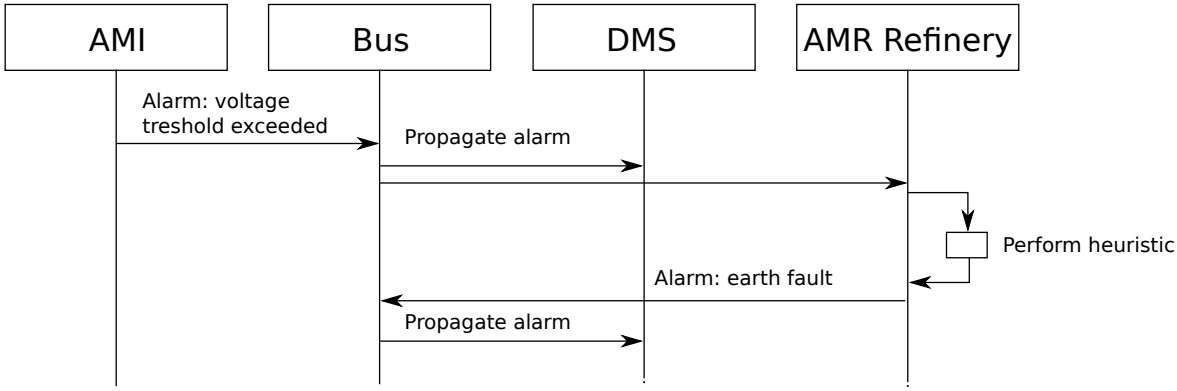

Figure 7.4: A proposed service which subscribes to AMR events and performs heuristics against a predefined set of rules. In this example the 'Refinery' service detects anomalies advertised by the AMI are likely caused by an earth fault and sends a more informative and higher priority alarm.

7.2.2 Workgroup positioning

The DMS 600 has the capability of overlaying positioning information from work groups on the field on the main map display of workstations. The current implementation works by having an adaptor to receive position information from a third party system and send change notifications on the internal DMSSocket protocol.

The next step in providing better situational awareness data would be to have the workgroups receive information from the control room operators, such as work instructions. Additionally, these teams would benefit from seeing the real-time state of the network on their end devices. Lastly, the teams should be able to communicate back the operations they have done.

These features would essentially require a two-way communication protocol between the field groups and the DMS system. Allowing these mobile clients to access the whole DMS system in most cases would not be acceptable, as the work groups connect through untrusted mobile networks, and possibly are subcontractors who should not have access to all of the data in the system.

7.2.3 Sharing outage data

The DMS 600 holds internally a comprehensive log of the network state, historical events and future plans. Sharing these to external stakeholders is a common requirement. Figure 7.5 illustrates the information requirements of different stakeholders. The image displays both real-time and periodical reporting requirements. While at first sight outage reporting and ongoing state seem to be separate concepts, both are renderings of the same data that should only be accessible internally by the DSO personnel.

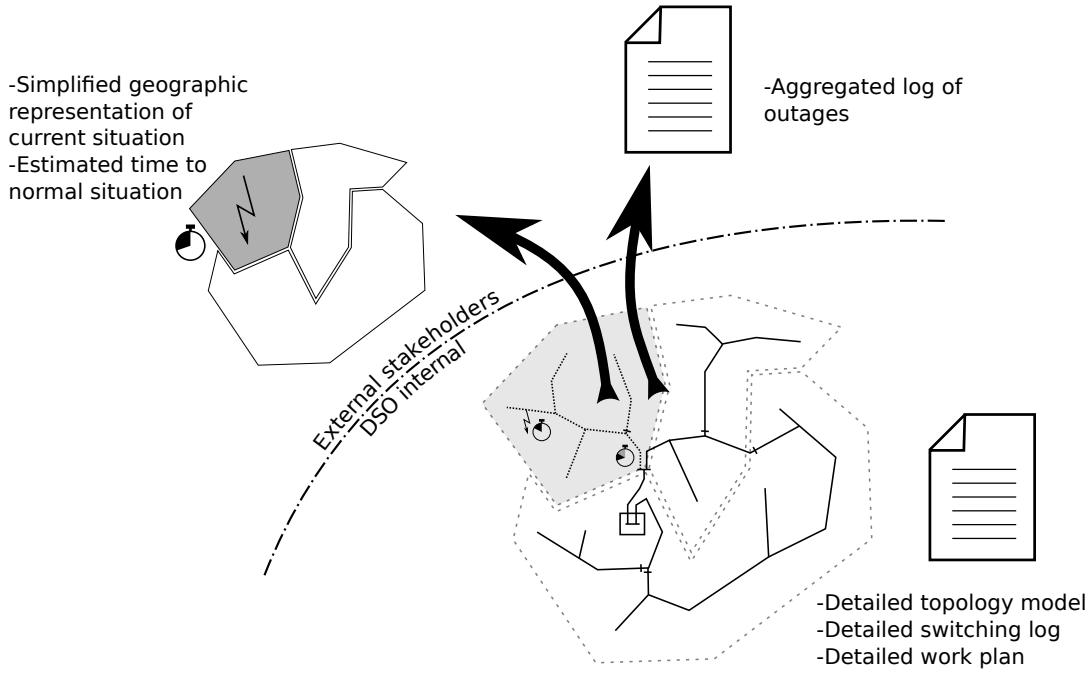

Figure 7.5: The relation of the internal data to the requirements of external stakeholders for both reporting and real-time situational awareness. Note that the internal data does not necessarily reside within a single vendor system.

One feature of the ABB DMS600 is producing data to a public web server to display ongoing and expected outages. The DMS produces periodically, typically every few minutes, an XML file with data on the outages. This data is parsed by a JavaScript application to display the data on a dynamic map. This design allows the service to be deployed on very simple web servers, as only one-way data is required. It also means no part of the DMS system has to be visible to the internet. The setup is illustrated in figure 7.6.

This same data may in future be leveraged by additional stakeholders, such as emergency response departments in cases of large scale disturbances in the network. A finnish research project aims to further develop the methods for sharing emergency information. This work is discussed in [70]. The same data is also useful for other forms of presentation, among them automated SMS text message notifications to

select customers. One aspect of the text message output is that it is desirable to only have notifications sent to customers when a specific change happens and the situation is known to remain stable for some time. [71]

A set of a few simple XML files is, however, not suitable for sharing larger data sets. DSOs are expecting to allow broader web based customer service, and some already add information from systems external to the DMS, such as energy measurements. Sharing such data requires authenticating the user and showing data only relevant to them. An expected regulatory requirement in Finland is delivering customers historical data on outages that have affected them. For security and implementation reasons, a database is required on the web server, which only gives out relevant data to specific customers.

The expanded features could be implemented by batch exporting of a subset of the data from the main DMS database to the partially-internet-exposed database. Another service oriented approach is proposed here. The web server could have a service that listens to specific broadcasts on a bus and stores relevant data internally. This approach would allow greater compatibility between parts of the system from different vendors. A batch database transfer would require the implementation to be planned according to internal database structures of both the DMS and the web service. A CIM based service oriented architechture would also share the interface with possible other systems that take advantage of the same data. The proposed service layout is displayed in figure 7.7.

From the image it can be seen that two-way data transfer might now be required, if only to allow the relevant publish-subscribe traffic to be compatible with the implementation of the bus. This should not present security issues as long as the design of the bus is taking security in account from the beginning. Only messages defined as originating from the web service should be accepted. Only those that are specifically allowed for export should be sent to the web server, regardless of what the web server (possibly compromised) might request.

Outage reporting is a key feature of a DMS. With the system logging all switching operations and following the state of the network topology, it can provide exact data on outages: durations, customers affected etc. During outages, the control room operators add additional data such as outage classification which is stored with the other records.

The internal outage records are usually presented to the end users aggegated in a format specific to the DSO. Some of these reports follow national regulations guidelines, others follow company standards in place within the DSO. The reports access the internal data through a database interface.

Outage reporting has additional considerations compared to typical operations within the DMS system. The reports are often run by personnel not directly involved

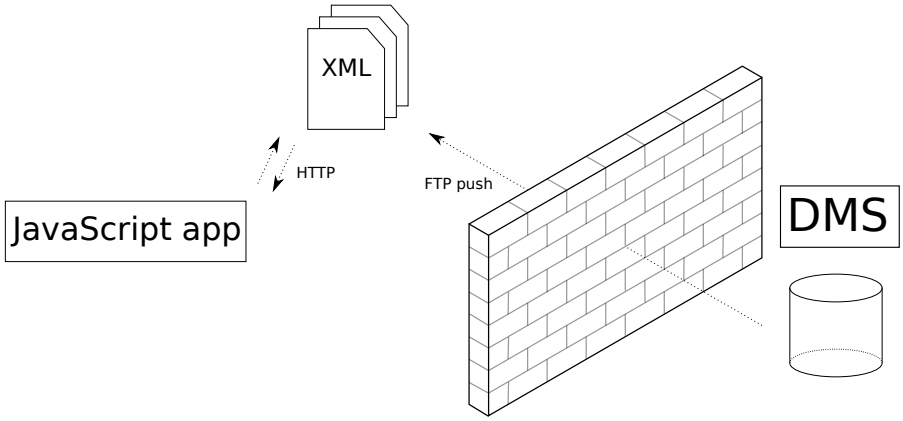

Figure 7.6: Current outage notification web service. Data on outages is pushed as an FTP file dump to the web server, where a JavaScript application accesses these static files. The DMS environment is heavily firewalled from the internet.

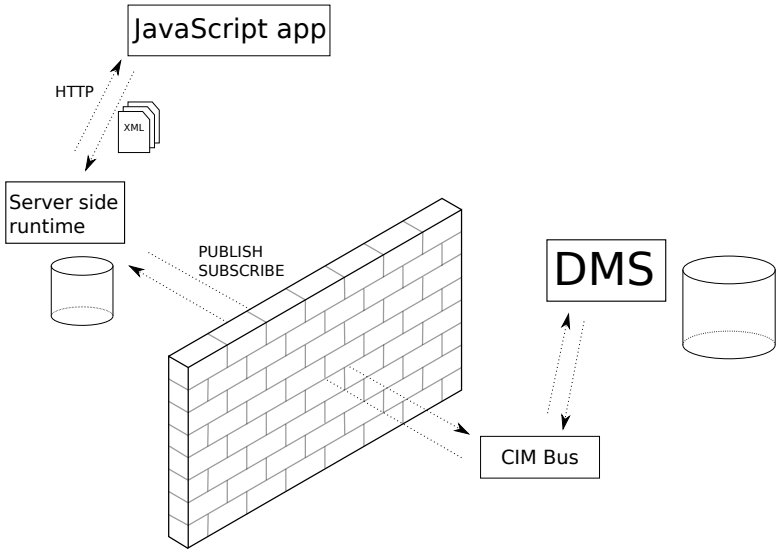

Figure 7.7: A possible implementation of a more feature-rich web based customer notification system.

in the system operation. It is desirable to be able to separate the reporting from the network where the actual system is run to allow a range of personnel access on aggregations of otherwise sensitive data.

In some deployments, the outage records have to be frozen after they are filed. Any modifications to the frozen records have to be logged and verified. This may be due to legal requirements or internal company guidelines.

When an outage record is created, notifications should be sent to multiple parties involved, such as customer service (to allow for informative replies to customer queries), personnel in charge of reporting (to ensure requred data is filled in by the operators as soon as possible) or management (to keep them updated on events).

These requirements: access control, separation from operation environment and up-to-date notification propagation makes many applications of the outage tracking a prime target for SOA-type integration.

8. CIM NETWORK MODEL FORMAT

As stated earlier, the CIM model provides a format for transferring network models. The recommended markup for transferring bulk network data is RDF. CIM objects are used to represent components. An RDF/XML syntax is defined in part 552 of IEC 61970 and is augmented by other parts of the standard. For partial updates over an event-based communication protocol, the standards allow the use of plain XML markup even for network data. An example of such case would be an update on the electrical data of a single component, transmitted over eg. a Web Service link.

A good network data import/export should be able to represent all data inside the software in an unambiguous way, all the while not adding implementation-specific types, unnecessary objects or enforcing semantics in naming conventions. As the internal model in DMS 600 differs at points greatly from that of the CIM, objects need to be generated and concatenated on each import/export run. The goal of getting standards compliant data out and generating the exact same network from this data is elusive, but CIM does seem to provide the tools to at least get close.

8.1 Topology

The topology model of CIM differs from the internal model used in DMS 600. CIM also allows for more variation in entities to represent the same data. The basic topology modelling block in DMS 600 is an MV Node, which is an entity that always has a geographical location and a type. An MV node can be eg. a transformer, a disconnector or a customer connection point. A special type of node, the branching node, is used to represent sitations where no component other than a connection is present in the topology.

MV Nodes are connected to each other by MV Sections. A section includes information about the conductor type, and a special conductor type, *BUSBAR*, is used when connections of negligible impedance are modelled.

In the CIM model, *ConductingEquipment* is a supertype of all network components. Each *ConductingEquipment* can have one or two *Terminals*, which represent connections to other components. Each *Terminal* is associated to exactly one *ConductingEquipment* and can connect to zero or one *ConnectivityNode*s, which is an object that connects the terminals with zero impedance.

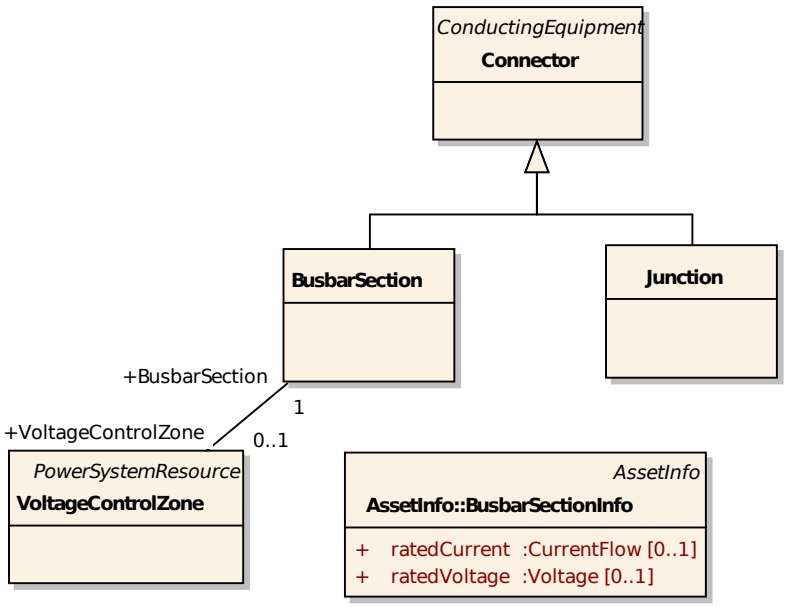

Figure 8.1: The CIM *Connector* classes.

Line sections are not represented by separate object types, instead they are subtypes of the same *ConductingEquipment* used to represent all components. Special objects, also subtypes of *ConductingEquipment* are used for representing busbars.

CIM provides two classes for describing connectors, *BusbarSection* and *Junction*. They are inherited from *ConductingEquipment* and used by connecting a single terminal to a *ConnectivityNode* - they can be thought of as kinds of descriptors to the *ConnectivityNode* instead of actual network components. The DMS600 models a connector using a branching node or as a separate busbar object. The inheritance of these types is shown in figure 8.1.

Figure 8.2 illustrates the DMS 600 data model, showing an example of components connected to each other. The same topology is represented in the CIM format in figure 8.3. Here the components are subtypes of *ConductingEquipment*, which links to a *Terminal* object, which in turn can connect to one object of type *ConnectivityNode*. The situation is more complicated in the CIM model when the connection is made with non-zero impedance. This is shown in figure 8.4. The topology model in DMS600 internal format is the same for both CIM examples, the only difference being the conductor type.

When translating from CIM to the DMS 600 internal format, the *ConnectivityNode* can either be neglected, or an MV Node of type *branching node* can be created to represent it. In the latter case, two separate MV Sections have to be generated. The additional node may make the model more complicated than is desired, so in the examples of figures 8.3 and 8.4, no additional node should be created. However, the CIM model allows for more than two connections to be made

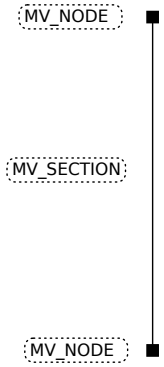

Figure 8.2: A basic topology in DMS 600 internal format. Two components are connected to each other via an MV Section. The electrical characteristics of the connection are represented by a conductor type associated with the MV Section. A special conductor type, *BUSBAR*, is used to represent connections with zero impedance.

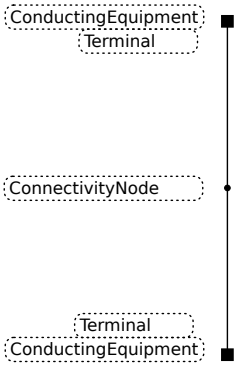

Figure 8.3: A basic connection between two components in the CIM data model. In this example, the components are connected with zero impedance, so in the DMS 600 model the *BUSBAR* conductor type would be used.

to the *ConnectivityNode*. In DMS 600, a section can only have two endpoints, so a *ConnectivityNode* with more than two connected *Terminals* would have to be modeled as a branching node. This situation is illustrated in figures 8.5 and 8.6.

Further differences in the network models are shown in figures 8.7 and 8.8. As can be seen, a major difference is in the handling of voltage levels. In DMS 600 low voltage networks are separate from medium voltage even at the level of data structures. In the CIM model, all components that affect the network topology are subclasses of *ConductingEquipment*. This has been left out from the picture to maintain simplicity. The DMS 600 handles two base classes, nodes and sections, while CIM only has *ConductingEquipment*.

8.1.1 Voltage levels

The internal data model of DMS 600 separates medium voltage and low voltage networks in different data types altogether. To work with low voltage networks, the

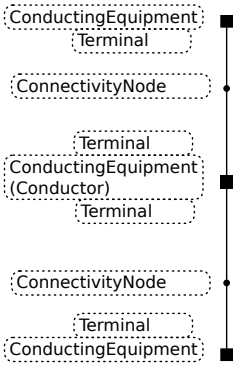

Figure 8.4: A connection between two components in the CIM model. Analoguous to figure 8.3, except here the connection has non-zero impedance and thus a *Conductor* object is used between the components. Note that *Conductor* is a subtype of *ConductingEquipment*, which is used to represent all other components.

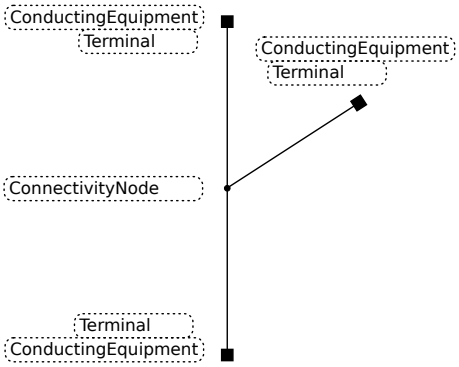

Figure 8.5: A CIM topology with multiple components connected to each other with zero impedance.

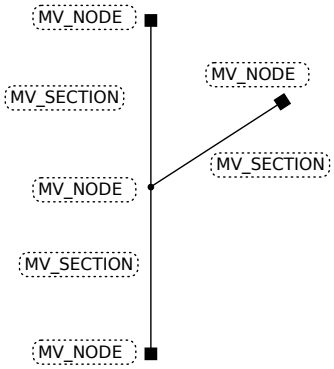

Figure 8.6: The DMS 600 topology analoguous to figure 8.5

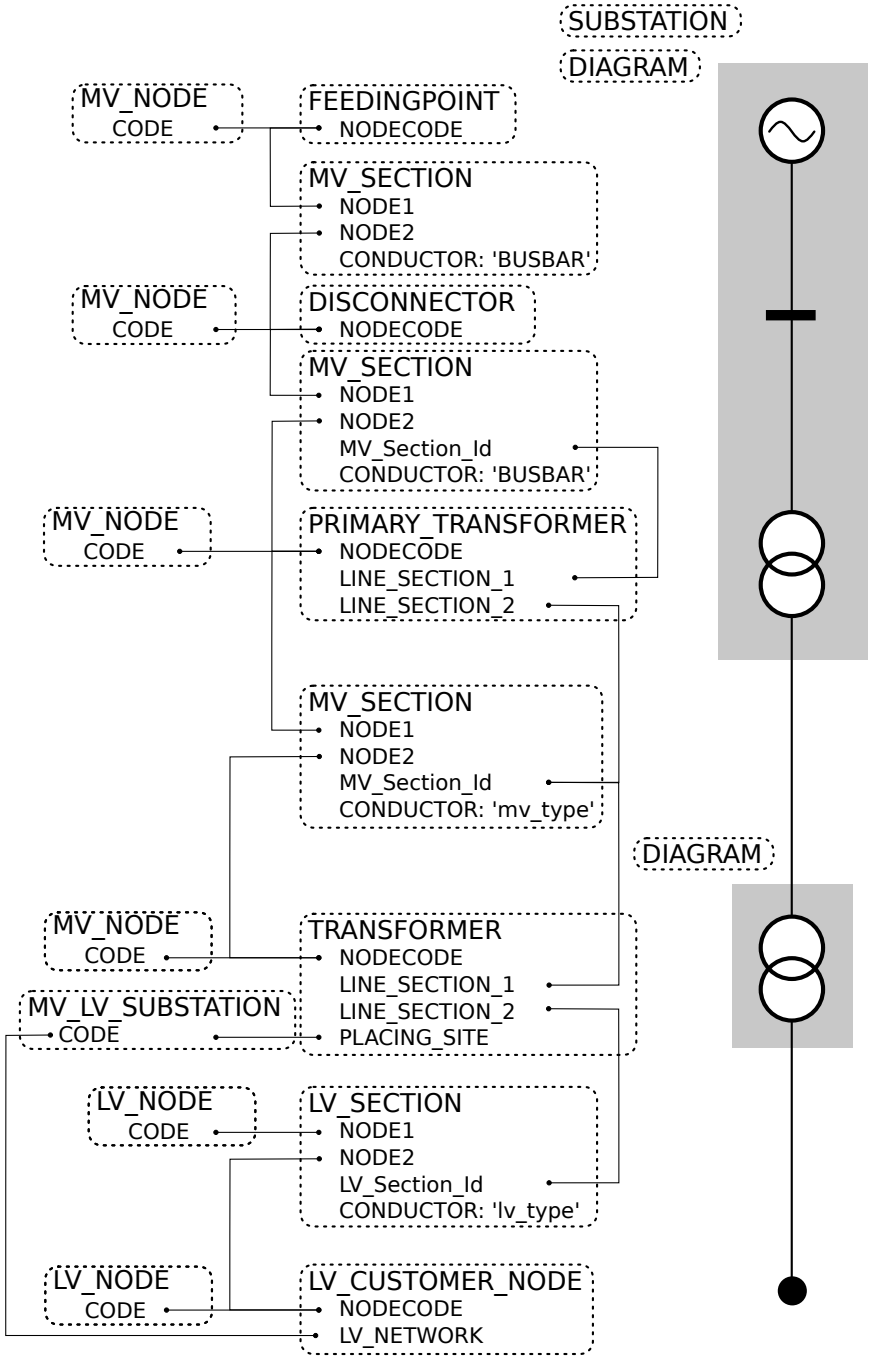

Figure 8.7: A more advanced network in the DMS 600 internal topology model.

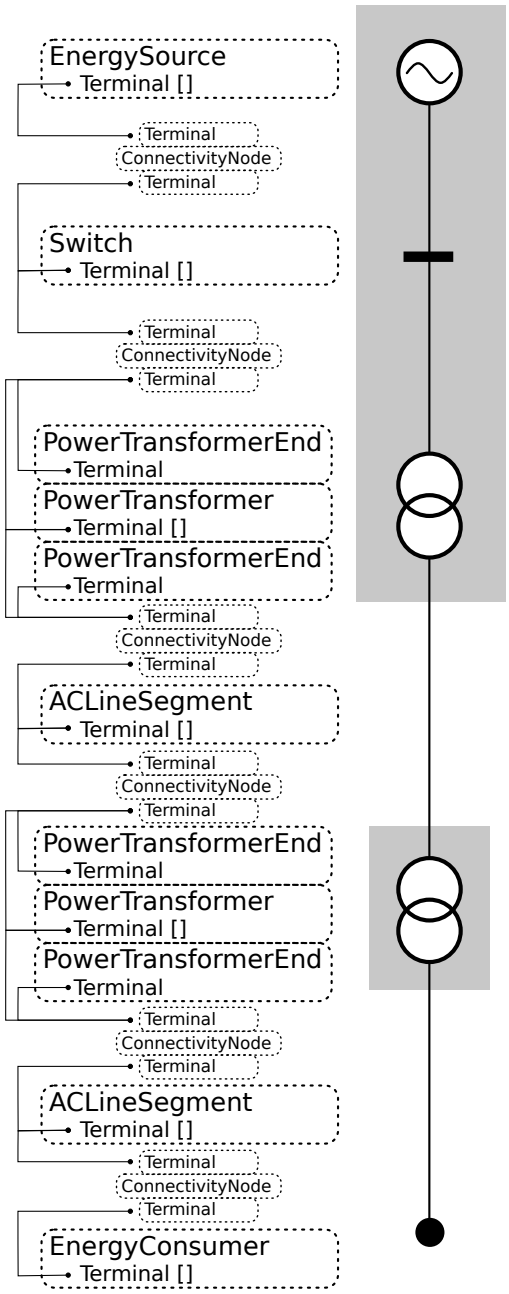

Figure 8.8: The network of figure 8.7 expressed as a CIM model.

user must explicitly load a portion of the low voltage network to memory. Commonly each distribution transformer feeds a certain part of the low voltage network. Even if low voltage networks of separate transformers have connecting lines, this is only assumed to be for providing a separate feed in case of transformer failure or maintenance and therefore each load point is assumed to statically link to a single transformer.

In the CIM model, low, medium and high voltage components use the same data types. There are no separate "LV Energy Consumer" and "MV Energy Consumer" objects, unlike DMS 600. When importing CIM network data to DMS 600, an arbitrary voltage must be selected as the crossover point upon which it is decided whether the import functionality should create low voltage or medium voltage objects. Some import functions also need to be duplicated, as DMS 600 low voltage types are not exact copies of medium voltage types.

Unfortunately, even this is not enough in all cases. It is possible that a CIM object does not define the *ConductingEquipment.BaseVoltage* property. In this case, the network topology needs to be traversed to determine the correct voltage level.

8.1.2 Transformer modelling

The basic transformer model has classes *PowerTransformer* and *PowerTransformerEnd*, where *PowerTransformer* is a subclass of *ConductingEquipment*. Each *PowerTransformerEnd* describes a single three-phase terminal of the transformer, so a typical transformer model would consist of one *PowerTransformer* and two *PowerTransformerEnd*s, one for the primary voltage and one for secondary. Although *PowerTransformerEnd* is not a *ConductingEquipment*, it associates to a single *Terminal* to allow describing which part of the network is connected to it.

Additionally, CIM provides classes *TransformerTank* and *TransformerTankEnd* to model the winding details for each phase. A *TransformerTank* objects associates with *PowerTransformer*. Both *TransformerTankEnd* and *PowerTransformerEnd* share a common parent, *TransformerEnd*, which allows for a shared representation of most of the electrical data, even though a transformer model should only contain either *TransformerTankEnd* or *PowerTransformerEnd*.

The CIM also provides the class *TransformerEndInfo* to allow a single data describing object to be associated to multiple transformers. This is useful especially in the case of distribution networks, where multiple transformers are of the same type and therefore share electrical data.

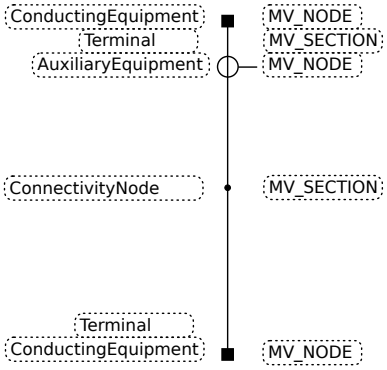

Figure 8.9: A current transformer as modeled in CIM and DMS600.

8.1.3 Auxiliary equipment and connectors

The CIM model treats auxiliary equipment such as fault indicators[1] and voltage[2]- and current[3] transformers differently from other components. While normally all components are subclasses of *ConductingEquipment*, these components are subclasses of *AuxiliaryEquipment*. Each *AuxiliaryEquipment* object only connects to exactly one terminal, while *ConductingEquipment* connect in most cases to two. Additionally, *AuxiliaryEquipment* do not have their own terminal, but they share one with a component of type *ConductingEquipment*. In the DMS600 model these components are ordinary MV nodes. When importing *AuxiliaryEquipment*, additional *BUSBAR* line sections need to be created when an *AuxiliaryEquipment* is connected to a terminal of a *ConductingEquipment*. The modelling difference is illustrated in figure 8.9.

When current and voltage transformer readings are directly used by a relay, they may not be modelled as network objects in the DMS, instead the electrical properties are part of the relay data sheet. To accurate create these models, additional logic would have to be implemented to decide whether a separate object is required or not. If, for example, a terminal of a *ProtectedSwitch* has a *PotentialTransformer* connected and the *OperatedByProtectionEquipment* defined, the potential transformer could be collapsed within the relay model.

8.1.4 Identifier consistency

As the internal DMS600 objects do not directly map to CIM objects in every case, new objects have to be dynamically created to represent the real situation. This step should ideally be deterministic, meaning every time the generated objects have the same properties and also have the same identifiers. To ensure this, it is foreseen

[1] *FaultIndicator*
[2] *PotentialTransformer*
[3] *CurrentTransformer*

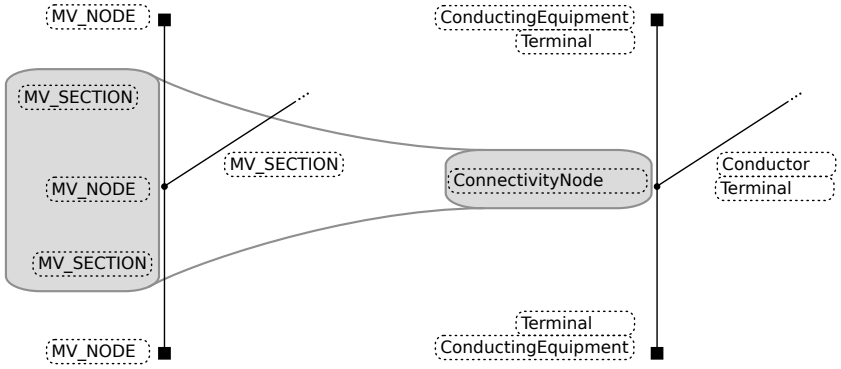

Figure 8.10: Possible data in lookup tables. The topology includes two components, connected to a branching node by the zero-impedance *BUSBAR* conductor type, and an additional conductor leaving the node with a non-zero impedance conductor. Enough data should be stored in a lookup table to ensure that this connection is collapsed to a single *ConnectivityNode* having the same identifier on every export run, and expanded into the same DMS objects on every import.

that some lookup tables need to be created in the database. An example of such a situation is illustrated in figure 8.10.

On import, it is enough to save the DMS identifiers of each node and section. One CIM object, e.g. the *ConnectivityNode* of the previous figure, may equal to multiple DMS 600 objects. The opposite is also true, the most basic example being the *ConductingEquipment-Terminal* pair that is represented by an MV node in DMS 600. During export, the same lookup tables should be used.

Unless identifiers are consistent, no incremental updates on the model can be performed. With fully consistent object creation, network data can be tracked up to the level of a single object, such as a network component.

As the DMS600 integrates with SCADA systems, it requires certain objects to define a SCADA code. Often this would be an OPC code. The CIM standard has no explicit field to define an OPC code, so the class *Name* should be used. Each incoming file should use a predefined *NameType* to identify a *Name* as an OPC code.

8.1.5 Coordinates

The DMS600 is fundamentally a lightweight GIS system. In the DMS600 data model, every component that can possibly have a coordinate associated with it, has to have it. This is essential for drawing graphics and visualizing the network. A CIM network data file on the other hand may be generated for a multitude of purposes. For calculations, it is possible to use a data file with no geographic information. Even if some coordinates are provided, another system may only provide them for those parts where they clearly can be determined from the real world, such as line sections.

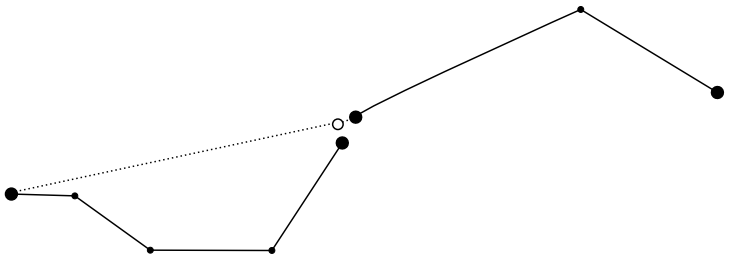

Figure 8.11: Failing creation of busbar section from connectivity nodes when *Terminal.SequenceNumber* is missing or wrong. Dashed line represents a zero-impedance line section, represented as conductor type *BUSBAR*.

So if a circuit-breaker and a primary transformer sit within a substation, it may be reasonable to provide the coordinates of the substation and just assume the other components are within that point or area. In order to import such data as a usable model in DMS600, some coordinates have to be automatically generated based on the data available. If no coordinate information is present, there is no way to do this unambiguosly. In implementations of the network import, the computer-generated coordinate information should be flagged as unreliable to prevent it conflicting with data in another GIS system.

Additionally, a single CIM model may contain coordinates in many formats. Each *Location* references a *CoodinateSystem* object, and different components may provide different coordinates, for example coordinates in separate mercator projections. For further complication, an additional *Diagram* object exists in CIM. It can be used to specify abstract coordinates and even a separate polygon representation for anything of type *IdentifiedObject*. Multiple diagrams may reference a single model object. The DMS600 does use a similar system of diagrams for substation layouts, where the diagrams are obtained from SCADA, but each diagram when imported gets defined corner points as real-life coordinates and is always drawn in that location. In DMS600, only a single diagram can reference a single power system object.

For line sections, a list of coordinates is given. For network calculations the order of coordinates naturally does not matter[4], so not all systems include the sequence numbers in the network model. However, for a visual representation of the topology, the order of the coordinates is more important. The order of coordinates should correspond with the order of terminals to ensure there are no extra-long zero-impedance sections that connect wrong terminals graphically. Thus, at least all *Conductor* objects should define the terminal order. The results of failure to do this are illustrated in figure 8.11.

[4]except in the special case of a line section including a *Cut* object, which has the property *Cut.lengthFromTerminal1*.

8.1.6 Conductor Types

The DMS600, being oriented for distribution systems, requires all line sections to reference a single conductor type. The CIM model, however, allows an *ACLineSegment* to either define the eletrical properties of the whole section or in the case of a DCIM[5] model, reference a *WireInfo* object that defines the electrical properties in per-length form. If incoming data uses the former method, DMS600 Conductor types must be created accordingly. While this is possible, it easily leads to several very similar conductor types. Therefore some additional heuristics may be needed to determine whether two *ACLineSegment* objects have 'close enough' electrical datas to use a shared conductor type.

Even if sufficient *WireInfo* objects exist in the model, the DMS600 wire model does not directly map to the CIM model. The CIM *WireInfo* and related classes are shown in figure 8.12 and the conductor type definition of DMS 600 in figure 8.13. In general, the CIM model provides more fine-grained information than the DMS model. Some properties are needed by the DMS600 that are not provided by the standard CIM, such as the 1 s short circuit current (for protection analysis), the cooling time constant τ (for analyzing acceptable time-dependent overloads for cables) or conductor mass and cost (for planning analysis). Some of these could be deduced from the cable properties, but doing so would open up many possibilities of miscalculation for properties that are often standard datasheet material for conductor vendors.

The conductor model differences between CIM and DMS 600 provide a good example of the problems in implementing a lossless round-trip network model import/export functionality. The cooling time constant is used to calculate cable cooling time using the equation

$$\Delta T(t) = \Delta T_0 e^{\frac{-t}{\tau}} \tag{8.1}$$

where ΔT is the ambient-cable temperature difference and t is time. When a cable is subjected to multiple overload situations on a short cycle, the subsequent overloads run a higher risk of heating the cable above the allowed limit. This is illustrated in figure 8.14. The time constant is sometimes estimated using the equation

$$\tau(min) = \frac{1}{60}\left|\frac{I_{1s}}{I_{max}}\right|^2 \tag{8.2}$$

where I_{1s} is the maximum allowable one second current and I_{max} is the maximum allowable continous current. The exact time constant is defined as

[5] CIM with Distribution extensions

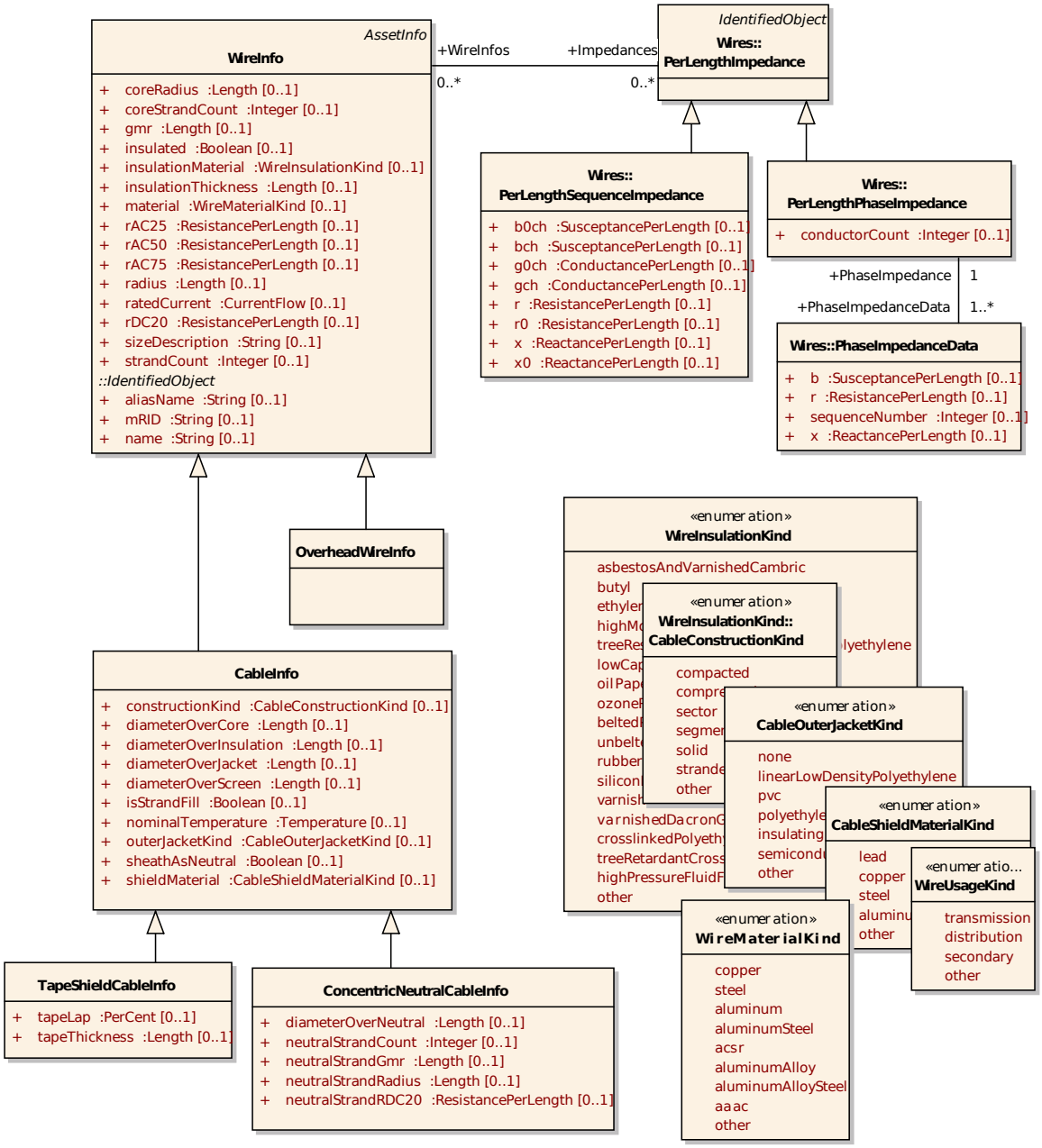

Figure 8.12: The CIM *WireInfo* class with direct relations and child types. Note that the *PerLengthImpedance* can be specified either by symmetric components (*PerLengthSequenceImpedance*) or as an impedance matrix (*PerLengthPhaseImpedance*), so methods should exist to use either model in calculations. The current DMS 600 model utilizes mainly a symmetric component model.

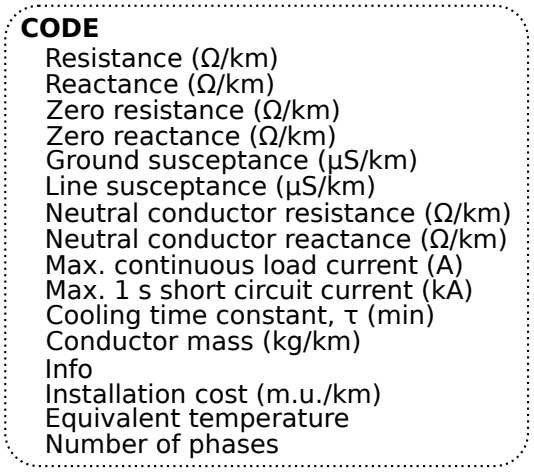

Figure 8.13: The DMS600 internal model for conductor types.

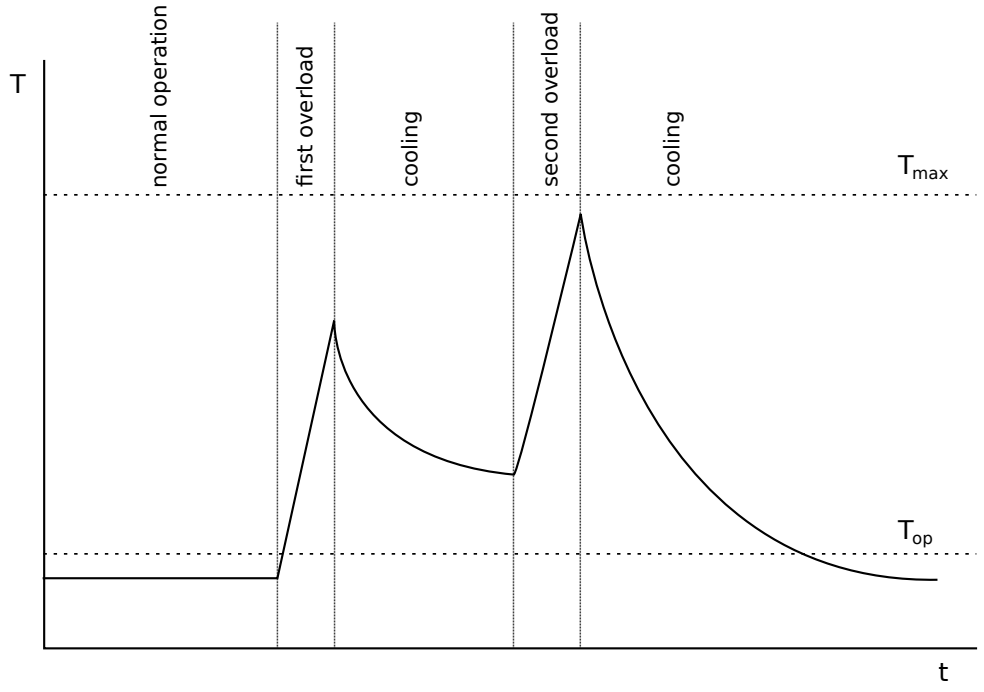

Figure 8.14: Cable temperature plotted against time in a situation with multiple subsequent overloads. The cooling slope is defined by the thermal time constant (τ) of the cable.

$$\tau = \frac{\rho c_p V}{h A_s} \tag{8.3}$$

where ρ is the density, c_p is specific heat capacity, V is volume, A_s is the surface area and h is the heat transfer rate. This means that to actually come up with the correct τ, one would also need to take into account the heat transfer rate. Heat transfer depends on the temperature gradient, and materials used. The IEC 60287-2 gives formulas for estimating thermal resistance of conductors, which is related to heat transfer rate by the equations

$$h = \frac{\Phi}{A_s \theta} = \frac{1}{A_s T} \tag{8.4}$$

where Φ is the heat flux, θ the temperature difference and T the thermal resistance. [72]

According to IEC 60287-2, the thermal resistivity of surrounding ground varies greatly depending on the type of ground, where generally more moist ground is better at transferring heat. The recommendation is to use values measured in similar conditions. [73]

These equations mean that the CIM model is sufficient for an engineer to make rudimentary network calculations, but importing the data to an existing system is not as straightforward. In this case, the import routine could use a combination of a materials- and installation types database and cable data to come up with a value for τ.

8.2 Differences between transmission and distribution networks

The modeling requirements for transmission and distribution networks vary slightly. In a real-world use case, distribution networks need to model a greater amount of components such as line sections and transformers. In a transmission network, the properties of individual objects are known to a much greater detail. In distribution networks however, the only available data may be type information which is then assumed to be applicable to a large number of components.

The CIM model addresses this difference by adding certain object types for describing type information in part 11 of IEC 61968. This documents builds upon part 452 of IEC 61970, which defines the basics of representing network topology. The base class for these type definitions is *AssetInfo*, which can relate to any object derived from type *Asset*. Additional subclasses are defined for transformer, switch, wire and busbar specifications.

The operation of transmission networks also differs greatly from operation of dis-

tribution networks. In transmission networks, looped connections are used to ensure redundancy. Distribution networks are usually designed to be operated in a radial configuration, where load flow direction is clearly from substation to customer nodes. A looped state is an anomaly and load flow calculations are usually limited in this mode of operation due to the much greater complexity of the network. These differences need not be represented in the data models, but when considering software packages to run in different environments, it must be noted that a calculation algorithm designed for one mode of operation may have performance problems or even be completely incompatible in another. A generalization is that transmission network algorithms use simple data in complex calculations while distribution networks use complex data in simple calculations.

A specific complication in distribution networks is the possibility of single-phase installations. Many distribution systems only have balanced three-phase installations, but in some countries this may not be relied on. The DMS600 recently added support for single phase systems, and the distribution extensions of the CIM allow the modeling of these.

8.3 Implementation of network data import

To successfully communicate event data in the style of a service oriented architechture, in many cases the counterparties need to have consistent topology models. This is especially important in cases such as the exchange of measurements or calculation results. The DMS 600 already has a proof-of-concept level implementation of network data export to the CIM/RDF format. During the work for this thesis, work on network model import functionality was started.

For event data, where the IEC standards recommend using basic XML markup, libraries and development tools are readily available to serialize data to the correct XML format, using a computer-generated XML Schema file as the reference. For the RDF format, tools are more sparse. Therefore, a simple RDF serializer/deserializer was developed. It uses C# code generated from the XML Schema as reference, and reads in RDF data to create internal in-memory instances of the CIM classes. The serializer heavily leverages dynamic typing capabilities of the C# language. Theoretically, in the event of a change in the CIM model, only code generation needs to be run and the serializer will adapt to changes. This also means that if multiple incompatible CIM versions need to be supported, multiple branches of generated code have to be maintained.

To ease the translation from CIM objects to DMS internal objects, a database serializer was also implemented. It dynamically creates database tables with the input RDF data filled in. The idea is to run simple SQL queries over these tables and insert the results in the DMS 600 internal database tables. This approach

works well for simple translations, such as network component models. For topology translation, the SQL approach is not sufficient to allow for maintainable code as similar CIM models may need to be translated into very distinct DMS 600 models. Some supporting code was implemented in the C# language to allow for a consistent and robust translation capability.

Another benefit of the approach of copying all input data to the database is that a consistent reference is created to the original data. In case the DMS 600 network model does not support all of the input data, the data is at least stored in an usable form. Possibly a future version the DMS 600 can directly reference the CIM objects where appropriate, ie. where new functionality is developed in the DMS 600 which uses data formats similar to those already defined in the CIM model.

For upgrading the previous export functionality, the code developed for this project may be leveraged. The same database and RDF serializer classes already allow for a reverse translation. To update the export functionality to a recent CIM version, some SQL queries or other code would need to be developed to fill in the CIM database tables from the DMS 600 internal tables.

The general architechture of the import functionality is shown in figure 8.15. As the functionality heavily leverages code generation, updates to the CIM model only require updates in the SQL conversion scripts. All other functionality can be updated by simply running code-generation against the new model definition. The use of an XML Schema as an intermediate code generation step ensures that functionality using plain XML markup, such as web service implementations, uses the exact same data model as the RDF functionality. The downside is that some semantic validation provided by RDF Schema definitions is not possible.

The network import functionality was mainly developed and tested based on the CIMUG Enterprise Architect UML models and some files produced by other vendors for CIMUG hosted interoperability tests. Eventually only one real-world external model was usable in an unmodified state, due to the others lacking geographic data, which is needed for the DMS to achieve its primary function as a situational awareness tool. More models are available in older formats, but the implementation used a draft version 15 CIM model and supporting older models at this stage was not deemed worthwhile. Future modifications are to be expected as the CIM model refines further.

The performance of the import functionality deserves a mention, as performance concerns are sometimes raised regarding the use of XML. This implementation was not optimized for speed. A small single-substation medium voltage network of around six thousand CIM objects takes around half-an-hour to import on a modern laptop computer running a database server locally. However, the main cause of the slowness is the use of single-shot database connections throughout the import pro-

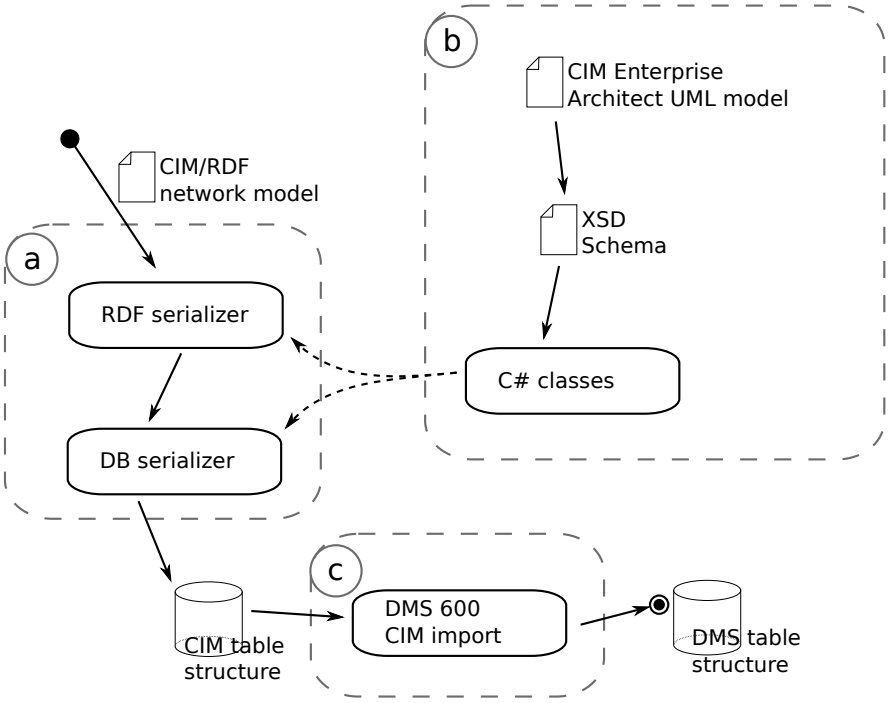

Figure 8.15: The general architecture of the import functionality. The components in the diagram are divided into three blocks depending on their maintenance requirements. Components of block *a* are transparent to CIM model changes. The operations of block *b*, creating the C# class representation of the CIM model are done on every update to the underlying CIM model, but require minimal developer intervention. The only component requiring maintenance to support new CIM model releases is the import functionality on block *c*, which consists mainly of SQL queries. The DB serializer and DMS 600 CIM import components can be reused when transferring CIM data from other formats, such as plain XML or a web service.

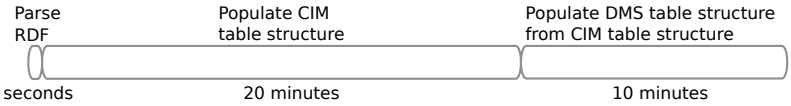

Figure 8.16: A breakdown of the processing time of a small RDF file on the current implementation.

cedure. While database access is generally a very fast way of transferring data, the overhead of opening a new query is relatively huge when transferring small amounts of data, therefore leaving a lot of room for optimization on the implementation. The actual parsing of the five megabyte XML file takes less than ten seconds. This does not undermine the fact that XML transformations are often identified as the performance bottleneck. This application does not perform such operations on the data. A coarse breakdown of the processing time is shown on figure 8.16.

Some improvements to the CIM/RDF export functionality were considered during the course of this thesis. Although an old proof-of-concept exists, changes in the CIM model have created a need for an updated version. The previous implementation did not leverage any code-generation or XML/RDF libraries, so instead of modifying that code a new implementation could be started, based on the work done on the export. The implementation should leverage the same lookup tables as the import to ensure consistency of identifiers up to a certain point.

8.4 Model and profile maturity

Barring minor exceptions such as the thermal properties of conductors, the CIM base model seems well suited for exchanging all types of information used by the DMS 600 system. Most cases of lack of necessary information on models tested come down not the base model missing properties, but the files not declaring them. This is a problem of either incomplete profiles, missing support for profiles by vendors, or both.

The current official CIM profiles are defined in the standards IEC 61970-501 and IEC 61968-13, the former concentrating on transmission networks and the latter on distribution. In light of the experiences from this thesis, the distinction between transmission and distribution profiles may not be as relevant as the different requirements between different types of information systems.

While calculations can be run on network models without any geographic information, a key feature of the DMS 600 is the visual representation - both geographical for objects than can definitely be placed on a map and pseudo-geographical diagram positions for objects in one place, such as within a substation. Thus a successful transfer of a model to DMS 600 would require comprehensive geographical data and additional diagram objects to place everything in a visually representative single

view. This can be achived through the CIM base model, but current CIM profiles do not seem to enforce it.

9. CONCLUSIONS

This thesis concentrated on two main topics. On the other hand the concept of Service Oriented Architecture was evaluated, given the CIM standards are expected to facilitate creating near plug-and-play interfaces for common tasks in the DSO computing domain. The maturity of the CIM network model was analyzed by implementing an import functionality from the CIM/RDF file format to the DMS 600.

DMS 600 is not implemented according to the principles of SOA. However, it does use a database system in the backend to share data and it includes a synchronisation mechanism, the DMSSocket which together can be tapped to implement new functionality in a service oriented way: data and events currently in the system can be accessed by implementing separate modules instead of expanding the client side application code.

The development of DMS 600 would benefit from more strict segregation between user interface functionality and backend calculations. In effect, the DMS600 client side code should be made thinner and more processing should move to services, which are easier to manage and oversee.

The CIM data types are already mostly in a usable state. Future changes are still expected to many parts of the model, so creating an interface based on CIM data types does not guarantee future compatibility with other systems. The CIM also does not in its current state define unambiguous service interfaces. For example, it is not possible to implement a calculation engine with a CIM interface and expect it to be plug-and-play compatible with another system. New interfaces, even if based on the CIM types, require some specification work.

The network model of the CIM is for the most part mature enough to be used for exchanging network data without additional input or user interaction. The main problem is that different types of systems, e.g. asset management and a DMS, concentrate on different information. The network data import that was developed as part of this thesis can currently create usable models from third party network definitions obtained from CIM User Group interoperability tests, but some polishing is required to make the model have all the information a typical DMS installation has. This is not an inherent problem with the CIM model, rather a problem of DMS requiring some properties on models that are not mandated by currently used CIM

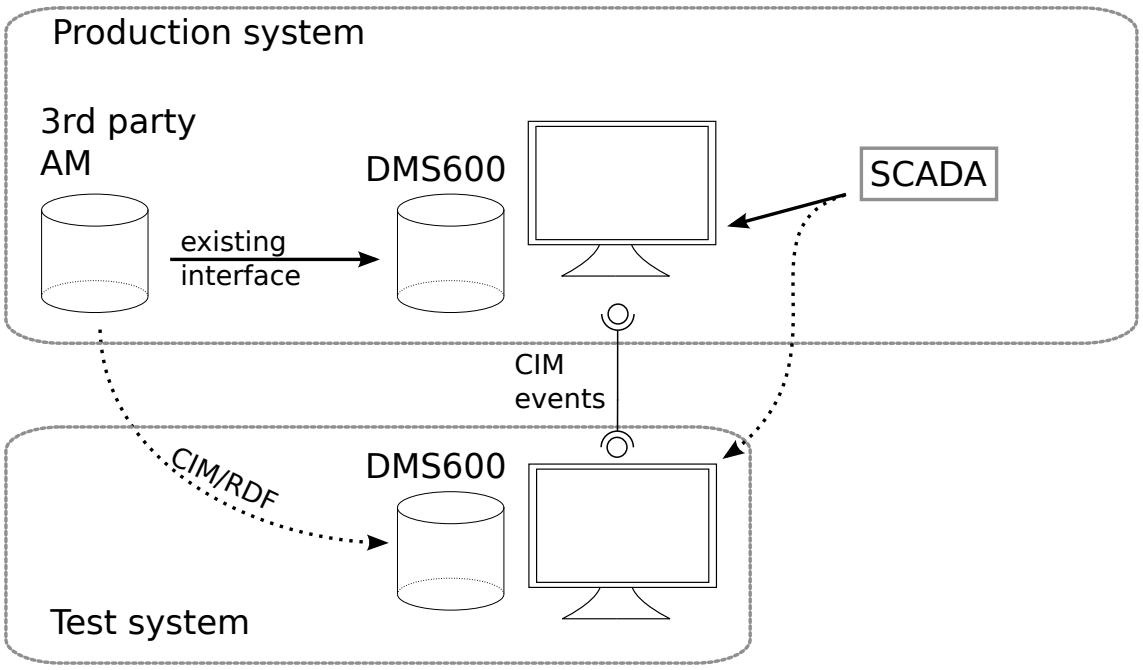

Figure 9.1: Testing the CIM/RDF import functionality on an existing system.

profiles.

The SOA techniques enable software vendors to deliver smaller pieces in a system, and this may create a new market for an integration consultant. The ABB project organisation is in a good position to enter this market, but some skills have to be acquired and maintained to successfully operate in this position. Primarily, understanding of the status of the CIM model and the types of data exchanges it enables is needed. Additionally, knowledge on the general technologies will be required, namely understanding of XML technologies, key WS-* standards and the concept of the message bus.

9.1 Future work

The CIM/RDF import functionality should be tested. A proposed way of doing this would be to set a testing environment on a site with a 3rd party asset management system that already has a proven interface with DMS 600 and a CIM/RDF export. In this kind of system, an initial network data import could be done on a clean DMS 600 installation to check whether the resulting network models match. The proposed test configuration is illustrated in figure 9.1.

Another benefit of setting up this kind of installation would be that it would ease further development of a CIM-based service interace. The two systems could be set to share event data only via the CIM interface and the test system would be monitored to check consistency with the production system. Although in this configuration both the receiving and sending ends of the interface would reflect the

ABB understanding of the standard, at least it would allow to pinpoint inconsistencies in the service implementation and events where sufficient data is not transferred on the CIM interface.

Currently setting up this kind of system is not possible as no publicly released CIM export functionality exists in the 3rd party asset management systems that already have an existing proprietary interface with the DMS 600. A simplified configuration would be to export data from an existing system and import it to a fresh one, thereby validating both the import and export implementations and proving they contain all the information available on a DMS 600 system.

BIBLIOGRAPHY

[1] E. Lakervi and E.J Holmes. *Electricity distribution network design.* IEE Power Engineering Series 21. Peter Peregrinus Ltd. on behalf of the Institution of Electrical Engineers, London, United Kingdom, 2nd edition edition, 2003.

[2] Sähkömarkkinalaki (Finnish law for electricity markets) 17.3.1995/386, 1995.

[3] M. Amin and B. F. Wollenberg. Toward a smart grid: power delivery for the 21st century. *Power and Energy Magazine, IEEE*, 3(5):34–41, 2005.

[4] A. Aminoff, I. Lappeteläinen, J. Partanen, S. Viljanen, K. Tahvanainen, P. Järventausta, and P. Trygg. Ostopalveluiden käyttö verkkoliiketoiminnassa. Technical report, Lappeenranta University of Technology, Tampere University of Technology, VTT, 2009.

[5] K. Channabasavaiah, K. Holley, and E. M. Tuggle. Migrating to a service-oriented architecture. Technical report, IBM, 2004.

[6] D. E. Cox and H. Kreger. Management of the service-oriented-architecture life cycle. *IBM Systems Journal*, 44(4):709 – 726, 2005.

[7] R. Welke, R. Hirschheim, and A. Schwarz. Service-Oriented Architecture Maturity. *IEEE Computer*, 2011.

[8] A. Arsanjani. Service-oriented modeling and architecture. Technical report, IBM, 2004.

[9] L. King. The Common Information Model for Distribution. Technical Update, Electric Power Research Institute, 2008.

[10] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, 2000.

[11] P. A. Laplante, J. Zhang, and J. Voas. What's in a Name? Distinguishing between SaaS and SOA . *IT Professional*, 10(3):46 – 50, 2008.

[12] H. Sirtl. Software plus Services: New IT- and Business Opportunities by Uniting SaaS, SOA and Web 2.0. 9 2008.

[13] J. P. Britton and A. de Vos. CIM-Based Standards and CIM Evolution. *IEEE Transactions on Power Systems*, 20(2):758 – 764, 2005.

[14] Tietoyhteiskunnan Kehittämiskeskus. `http://www.tieke.fi`, retrieved 12.11.2011.

[15] J. Partanen, S. Honkapuro, J. Lassila, T. Kaipia, P. Verho, P. Järventausta, J. Strandén, and A. Mäkinen. Sähkön toimitusvarmuuden kriteeristö ja tavoitetasot. Technical report, Energiatutkimus SER, 2010.

[16] Open Geospatial Consortium. About OGC. `http://www.opengeospatial.org`, retrieved 12.11.2011, 2011.

[17] A. Cox, P. Daisey, R. Lake, C. Portele, and A. Whiteside. *OpenGIS Geography Markup Language (GML) Implementation Specification*, 2004.

[18] J. de la Beaujardiere. OpenGIS Web Map Server Implementation Specification. Technical report, Open Geospatial Consortium Inc., 2006.

[19] National Rural Electric Cooperative Association. The MultiSpeak web site. `http://www.multispeak.org`, retrieved 12.11.2011.

[20] G. A. McNaughton and W. P. McNaughton. *MultiSpeak Version 3.0 User's Guide*, 2006.

[21] G. A. McNaughton, G. Robinson, and G. R. Gray. MultiSpeak and IEC 61968 CIM: Moving Towards Interoperability. 2008.

[22] V. Rasi. Common information model adaptation in power distribution management system environment. Master's thesis, Tampere University of Technology, 2009.

[23] CIMTool.org. `http://cimtool.org`, retrieved 12.11.2011.

[24] CIMSpy Enterpise Edition. `http://www.powerinfo.us/opensource/cimspyEE.html`, retrieved 12.11.2011.

[25] CIM EA, An Enterprise Architect Add-In for the IEC CIM. `http://www.cimea.org/`, retrieved 12.11.2011.

[26] DIgSILENT GmbH, Power System Engineering and Software. `http://www.digsilent.de`, retrieved 12.11.2011.

[27] Strategic review prepared for the LIPA board of trustees. Technical report, Long Island Power Authority, 2010.

[28] M. Hervey and P. Vujonic. A Common Language. *IEEE Power and Energy Magazine*, 2010.

[29] Interagency Report 7628, Guidelines for Smart Grid Cyber Security. Technical report, National Institute of Standards and Technology, 2010.

[30] Neetu Rajpal. *Using the XML Diff and Patch Tool in Your Applications.* http://msdn.microsoft.com/en-us/library/aa302294.aspx, retrieved 12.11.2011, 2002.

[31] diffxml - XML Diff and Patch Utilities. http://diffxml.sourceforge.net/, retrieved 12.11.2011, 2009.

[32] Oracle. Class reference, XMLDiff. http://otndnld.oracle.co.jp/document/products/oracle11g/111/doc_dvd/appdev.111/b28391/oracle/xml/differ/XMLDiff.html, retrieved 12.11.2011.

[33] A. de Vos. RDF Difference Models. Technical report, Langdale Consultants, 2002.

[34] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. *Extensible Markup Language (XML).* http://www.w3.org/TR/xml/, retrieved 12.11.2011, 2008.

[35] D. C. Fallside and P. Walmsley. *XML Schema Part 0: Primer.* http://www.w3.org/TR/xmlschema-0/, retrieved 12.11.2011, 2004.

[36] J. Clark and S. DeRose. *XML Path Language (XPath).* http://www.w3.org/TR/xpath/, retrieved 12.11.2011, 1999.

[37] J. Clark. *XSL Transformations (XSLT).* http://www.w3.org/TR/xslt, retrieved 12.11.2011, 11 1999.

[38] Stylus Studio XSLT Mapper. http://www.stylusstudio.com/xslt_mapper.html, retrieved 12.11.2011.

[39] *Mapping Between Schemas: BizTalk Mapper.* http://msdn.microsoft.com/en-us/library/ee253382%28v=bts.10%29.aspx, retrieved 12.11.2011, 2004.

[40] F. Manola and E. Miller. *RDF Primer.* http://www.w3.org/TR/rdf-primer/, retrieved 12.11.2011, 2004.

[41] D. Beckett. *RDF/XML Syntax Specification.* http://www.w3.org/TR/REC-rdf-syntax/, retrieved 12.11.2011, 2004.

[42] D. Brickley and R. V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema* . http://www.w3.org/TR/rdf-schema/, retrieved 12.11.2011, 2004.

[43] D. L. McGuinness and F. van Harmelen. *OWL Web Ontology Language Overview* . http://www.w3.org/TR/owl-features/, retrieved 12.11.2011, 2004.

[44] E. Prud'hommeaux and A. Seaborne. *SPARQL Query Language for RDF.* `http://www.w3.org/TR/rdf-sparql-query`, retrieved 12.11.2011.

[45] D. Beckett and J. Broekstra. *SPARQL Query Results XML Format.* `http://www.w3.org/TR/rdf-sparql-XMLres/`, retrieved 12.11.2011.

[46] K. G. Clark, L. Feigenbaum, and E. Torres. *SPARQL Protocol for RDF.* `http://www.w3.org/TR/rdf-sparql-protocol`, retrieved 12.11.2011.

[47] M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).* `http://www.w3.org/TR/soap12-part1/`, retrieved 12.11.2011, 2007.

[48] M. Gudgin, M. Hadley, and T. Rogers. *Web Services Addressing 1.0 - Core.* `http://www.w3.org/TR/ws-addr-core/`, retrieved 12.11.2011, 2006.

[49] D. Davis, A. Karmarkar, G. Pilz, S. Winkler, and Ü. Yalçinalp. Web Services Reliable Messaging (WS-ReliableMessaging). Technical report, OASIS, `http://docs.oasis-open.org/ws-rx/wsrm/200608/wsrm-1.1-spec-cd-04.html`, retrieved 12.11.2011, 2006.

[50] OASIS, Advancing open standards for the information society. `http://oasis-open.org`, retrieved 12.11.2011.

[51] K. Iwasa, J. Durand, T. Rutt, M Peel, S. Kunisetty, and D. Bunting. Web Services Reliable Messaging TC WS-Reliability 1.1. Technical report, OASIS, 2004.

[52] K. Lawrence, C. Kaler, A. Nadalin, C. Kaler, R. Monzillo, and P. Hallam-Baker. Web Services Security: SOAP Message Security 1.1 (WS-Security 2004). Technical report, OASIS, 2006.

[53] S. Shirasuna, A. Slominski, L. Fang, and D. Gannon. Performance Comparison of Security Mechanisms for Grid Services. Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04), 2004.

[54] A. S. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, and Ü. Yalçinalp. Web Services Policy 1.5 - Framework. Technical report, W3C, `http://www.w3.org/TR/ws-policy/`, retrieved 12.11.2011, 2007.

[55] Anthony Nadalin Marc Goodner Martin Gudgin Abbie Barbir Hans Granqvist Kelvin Lawrence, Chris Kaler. WS-SecurityPolicy 1.2. Technical report, OASIS, 7 2007.

[56] K. Lawrence, C. Kaler, A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, and H. Granqvist. WS-Trust 1.3. Technical report, OASIS, http://www.computer.org/portal/web/csdl/doi/10.1109/GRID.2004.50, retrieved 12.11.2011, 2007.

[57] E. Newcomer, I. Robinson, M. Feingold, and R. Jeyaraman. Web Services Coordination (WS-Coordination). Technical report, OASIS, http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os/wstx-wscoor-1.2-spec-os.html, retrieved 12.11.2011, 2009.

[58] E. Newcomer, I. Robinson, M. Little, and A. Wilkinson. Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.2. Technical report, OASIS, http://docs.oasis-open.org/ws-tx/wstx-wsat-1.2-spec-os/wstx-wsat-1.2-spec-os.html, retrieved 12.11.2011, 2009.

[59] D. Davis, A. Malhotra, K. Warr, and W. Chou. Web Services Enumeration (WS-Enumeration). Technical report, W3C, http://www.w3.org/TR/ws-enumeration/, retrieved 12.11.2011, 2011.

[60] D. Davis, A. Malhotra, K. Warr, and W. Chou. Web Services Metadata Exchange (WS-MetadataExchange). Technical report, W3C, http://www.w3.org/TR/ws-metadata-exchange/, retrieved 12.11.2011, 2011.

[61] Online community for the Universal Description, Discovery and Integration OASIS Standard. http://uddi.xml.org.

[62] Apache Axis2/Java. http://axis.apache.org/axis2/java/core/, retrieved 12.11.2011.

[63] Celtix: The Open Source Java Enterprise Service Bus. http://celtix.ow2.org/, retrieved 12.11.2011.

[64] Apache CXF: An Open-Source Services Framework. http://cxf.apache.org/, retrieved 12.11.2011.

[65] R. A. van Engelen and K. A. Gallivan. The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks. 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), Berlin, Germany., 2002.

[66] GlassFish - Metro. http://metro.java.net/, retrieved 12.11.2011.

[67] Windows Communication Foundation. http://msdn.microsoft.com/en-us/netframework/aa663324, retrieved 12.11.2011.

[68] Mono, Cross platform, open source .NET development framework. `http://www.mono-project.com`, retrieved 12.11.2011.

[69] P. Järventausta, P. Verho, S. Vehviläinen, A. Mäkinen, M. Kärenlampi, P. Trygg, K. Kivikko, T. Chrons, and A. Rinta-Opas. Using advanced AMR system in low voltage distribution network management. Vienna, 2007.

[70] H. Krohns, J. Strandén, P. Verho, and J. Sarsama. Developing communication between actors in major disturbances of the electric power supply. 2010.

[71] J. Kuru, J. Haikonen, and J. Myllymäki. Innovative system integration for outage communication. Prague, CIRED, 20th International Conference on Electricity Distribution, 2009.

[72] Martti Paavola. *Sähköjohdot.* WSOY, Porvoo, 1975.

[73] IEC 60287-2-1 Thermal Reistance - Calculation of thermal resistance, 2005.

APPENDIX 1: A LIST OF PUBLISHED AND PLANNED STANDARDS IN THE CIM FAMILY

The standards of the CIM family (IEC 61968 and IEC 61970) are listed in this appendix, along with their publication dates. Note that some information is publicly available also for the parts still in development, in the form of the EA model files from CIMug.

IEC 61968

Published	Part	Title
2003-10	1	Interface architechture and general requirements
2003-11	2	Glossary
2004-03	3	Interface for network operations
2007-07	4	Interfaces for records and asset management
Unpublished	5	Interfaces for operational planning and optimization
Unpublished	6	Interfaces for maintenance and construction
Unpublished	7	Interfaces for network extension planning
Unpublished	8	Interface standard for customer support
2009-09	9	Interfaces for meter reading and control
Unpublished	10	Interfaces for business functions external to distribution management
2010-07	11	Common information model (CIM) extensions for distribution
Unpublished	12	Common information model (CIM) use cases for 61968
2008-06	13	CIM RDF Model exchange format for distribution
Proposed	14	Mapping between MultiSpeak and CIM

IEC 61970

Published	Part	Title
2005-12	1	Guidelines and general requirements
2004-07	2	Glossary
Common Information Model (CIM)		
2009-04	301 (2nd ed.)	Common information model (CIM) base
Unpublished	302	Common information model (CIM) financial, energy scheduling and reservations
Component Interface Specification (CIS)		
2005-09	401	Component interface specification (CIS) framework
2008-06	402	Common services
2008-06	403	Generic data access
2007-08	404	High Speed Data Access (HSDA)
2007-08	405	Generic Eventing and Subscription
2007-08	407	Time Series Data Access (TSDA)
2008-06	453	CIM based graphics exchange
Unpublished	456	Solved power system state profiles
CIS Technology Mappings		
2006-03	501	Common Information Model Resource Description Framework (CIM RDF) schema
Unpublished	502-8	Web Services Profile for 61970-4 Abstract Services