TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

RISTO EEROLA

ANALYSING INTEGRATION AND INFORMATION SECURITY: ENTERPRISE SERVICE BUS SOLUTION FOR SMART GRID

Master of Science Thesis

Examiner: Professor Hannu Koivisto

Examiner and topic approved in the Automation, Mechanical, and Material Engineering Faculty Council meeting on 5th of December 2012

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master’s Degree Programme in Automation Technology
EEROLA, RISTO: Analysing Integration and Information Security: Enterprise Service Bus Solution for Smart Grid
Master of Science Thesis, 112 pages, 4 Appendix pages
June 2013
Major: Automation and Information Networks
Examiner: Professor Hannu Koivisto
Keywords: Smart Grid, integration, information security, Enterprise Service Bus, Microsoft BizTalk, Service Oriented Architecture, Common Information Model

Electricity is the lifeline of modern society. Without major improvements and new technology, the current electric grid cannot meet the future demand for safe, reliable, sustainable, and affordable electricity. A proposed solution is the Smart Grid that utilises advanced information and communication technologies (ICT). The Smart Grid will help to change the ways electricity is produced and consumed. This thesis focuses on two important areas in the Smart Grid: the integration of existing and new information systems, and the information security of the integration solutions.

The Smart Grids and Energy Markets (SGEM) is a project for extensive research on the future of electric energy. As part of the SGEM project, this thesis focuses on the integration of information systems within the distribution domain. Earlier research suggests that concepts such as Service-Oriented Architecture (SOA), Enterprise Service Bus (ESB), and Common Information Model (CIM) are essential for a successful Smart Grid integration. The goal of this work was to study these topics and to provide an integration component to be used in a concrete demonstration environment.

The theoretical background section consists of research on various integration architectures and their characteristics, and provides details of their functionality and performance. The integration landscape includes an introduction to the Smart Grid, the electricity distribution domain and related information systems, and the most important standards in the field. An introduction is provided to Microsoft BizTalk Server, the integration platform used in this project. Information security is a key aspect that cross-cuts the entire work. A specific section for related information security aspects is included for each of the discussed topics.

The experimental part of this work started from an example ICT architecture and three use cases as described previously within the SGEM project. The use cases are analysed in detail using a data flow approach to define the specific integration and information security requirements. A BizTalk based demonstration environment was designed and implemented. It will serve as a foundation for future work and allow for the integration of other parts of the example architecture.

The main result of this work is that, although SOA, ESB, and CIM are beneficial concepts, they are no silver bullet for integration issues. Further, they fundamentally change the approach to information security; this is particularly true for service-orientation. BizTalk offers a viable platform for integration, but, as an ESB, has certain limitations that must be carefully considered. A guideline for implementing the said concepts is offered to aid future integration work. It can be used to lower the barriers for collaboration between experts in the fields of electricity, integration, and information security. Co-operation of the foresaid parties is crucial for building secure, reliable, and efficient integration that will meet the needs of the Smart Grid.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Automaatiotekniikan koulutusohjelma
EEROLA, RISTO: Analysing Integration and Information Security: Enterprise Service Bus Solution for Smart Grid
Diplomityö, 112 sivua, 4 liitesivua
Kesäkuu 2013
Pääaine: Automaatio- ja informaatioverkot
Tarkastaja: professori Hannu Koivisto
Avainsanat: Smart Grid, älykäs sähköverkko, integraatio, tietoturva, palveluväylä, Enterprise Service Bus, Microsoft BizTalk, palvelukeskeinen arkkitehtuuri, Common Information Model

Sähköenergia on elintärkeää modernin yhteiskunnan toimivuudelle. Tulevaisuudessa tarvitaan yhä enemmän turvallista, luotettavaa, ympäristön kannalta kestävää ja riittävän edullista sähköenergiaa. Nykyinen sähköverkko vaatii kehittämistä ja merkittäviä parannuksia, jotta se pystyy vastaamaan näihin tarpeisiin. Ratkaisuksi on ehdotettu älykästä sähköverkkoa, Smart Gridiä. Tavoitteena on kehittää uusia tapoja tuottaa ja kuluttaa sähköä hyödyntämällä sähköverkon toteutuksessa laajamittaisesti tieto- ja viestintäteknologioita. Tässä työssä käsitellään kahta Smart Gridin kannalta tärkeää aihetta: tietojärjestelmien integrointia ja tietoturvallisuutta.

Smart Grids and Energy Markets (SGEM) -projekti tutkii laaja-alaisesti sähköenergian tulevaisuutta. Osana SGEM-projektia tämä diplomityö keskittyy sähkön jakeluverkon hallinnassa käytettävien tietojärjestelmien integrointiin, sekä siihen liittyvään tietoturvaan. Aiemman tutkimuksen perusteella integraatioratkaisun tärkeimmiksi osa-alueiksi on todettu palveluväylään perustuva palvelupohjainen arkkitehtuuri, sekä kaikille toimijoille yhteinen tietomalli. Tämän työn tavoitteena on tarjota konkreettisia ohjeita ja esimerkkejä mainittujen konseptien hyödyntämisestä. Tarkoitus on demonstroida projektissa aiemmin esitettyä malliarkkitehtuuria rakentamalla testiympäristö ja toteuttamalla siinä tarvittava integraatioratkaisu.

Yhtenä päätavoitteena oli tutkia integraation teoriaa ja eri arkkitehtuureja ja esitellä niiden toiminnallisuuden ja suorituskyvyn olennaisia eroja. Monet tahot tarjoavat ohjelmistoalustoja, jotka toimivat eri integraatioarkkitehtuurien käytännön toteutusten pohjana. Toinen päätavoite oli evaluoida erästä integraatio-ohjelmistoa, Microsoftin BizTalk Serveriä. Evaluoinnin pohjana ovat yksityiskohtainen analyysi ja BizTalkiin perustuvan demonstraatioympäristön rakentaminen. Tavoitteena oli toteuttaa tässä ympäristössä yksinkertaisia testejä ja luoda perusta, jota voidaan hyödyntää tulevissa testauksissa. BizTalk-ympäristön tulee mahdollistaa uusien järjestelmien integrointi myöhemmin. Tietoturva tulee ottaa huomioida integrointiprosessin kaikissa vaiheissa. Se on siten koko työtä läpileikkaava aihealue, jota erityisesti painotetaan.

Työn ensimmäinen osa esittelee teoreettista taustaa ja toimintaympäristön. Toinen luku esittelee lyhyesti sähköverkon toimintaa lukijoille, joilla ei ole sähköalan taustaa. Olennainen osa on älykkään sähköverkon tietoturva-aspektien käsittely. Smart Grid on ympäristönä ainutlaatuinen yhdistelmä perinteisen tietotekniikan ja automaatioalan järjestelmiä. Laajuutensa ja monimutkaisuutensa vuoksi se on ennennäkemättömän haastava toimintaympäristö tietoturvan kannalta. Automaatiojärjestelmien erityispiirteet, muun muassa reaaliaikavaatimukset, tulee huomioida myös tietoturvan suunnittelussa ja toteutuksessa.

Kolmannessa luvussa käsitellään integraation ja eri arkkitehtuurien kehitystä. Luvussa esitellään työn kannalta olennaiset konseptit: palveluorientoitunut arkkitehtuuri (Service-Oriented Architecture, SOA) ja palveluväylä (Enterprise Service Bus, ESB). Samalla käsitellään myös palveluväylän tärkeimmät erot perinteisempään yrityssovellusten integrointiin (Enterprise Application Integration, EAI) verrattuna. Väliohjelmiston (middleware) testaamiseen ja valintaan vaikuttavia asioita sekä tietoturvaa käydään läpi. Tietoturvassa erityisesti palveluorientoituneisuus aiheuttaa suuria muutoksia: monet perinteisessä sovellusarkkitehtuurissa käytetyt tietoturvan toteutusmenetelmät eivät enää ole käyttökelpoisia.

Neljäs luku esittelee aluksi tutkimusongelmaa ja toimintaympäristöä eli sähkön jakeluverkon moninaisia tietojärjestelmiä sekä niiden välisiä kommunikaatiotarpeita. Jakeluverkko-operaattorin (Distribution System Operator, DSO) tärkeimmät tietojärjestelmät sekä yhteinen tietomalli (Common Information Model, CIM) esitellään lyhyesti. Lisäksi tärkeimmät standardit ja suositukset käydään läpi, koska niillä on olennainen rooli minkä tahansa laajan ja monimutkaisen järjestelmän kehittämisessä. Tarkastelun näkökulmina ovat Smart Grid, integraatio yleisellä tasolla ja tietoturva Smart Gridissä. Lopuksi esitellään tietovuot ja tietovuokaaviot (Data Flow Diagrams, DFD), jotka tarjoavat hyvän perustan eri järjestelmien välisten tiedonsiirtotarpeiden käsittelyyn ja helpottavat myös tietoturvavaatimusten analysointia.

Työssä käytetty integraatioratkaisu, Microsoft BizTalk Server, esitellään viidennessä luvussa. Luvussa kuvataan lyhyesti, mitä BizTalk tekee, mihin sitä voidaan käyttää ja miten se on toteutettu teknisesti. BizTalk on pohjimmiltaan viestinvälitysohjelmisto (message broker). Viestien välityksen toteuttavien komponenttien ja toimintalogiikan esittely antaa hyvän kuvan BizTalkin toiminnasta ja käyttömahdollisuuksista. Toimintalogiikan lisäksi käydään lyhyesti läpi BizTalkin asennus, sovelluskehitys, ajonaikainen ympäristö ja ylläpito. BizTalk on kehitetty alun perin EAI-tuotteeksi, mutta ESB Toolkit -laajennuksen avulla sitä voidaan käyttää myös ESB-palveluväylän rakentamisen perustana. ESB Toolkitin kehitys ja toiminnallisuus käydään läpi. Lopuksi käsitellään myös BizTalkin tietoturvaominaisuuksia. Kuten monet väliohjelmistot ja integraatiotuotteet, BizTalk on monimutkainen ohjelmistokokonaisuus. On syytä korostaa, että sen syvällinen tuntemus vaatii huomattavaa kokemusta. Yhden diplomityön puitteissa BizTalk voidaan esitellä vain pintapuolisesti.

Työn toinen osa kuvaa esimerkkiarkkitehtuurin, rakennetun testiympäristön ja testauksen pohjana toimineet kolme käyttötapausesimerkkiä. Arkkitehtuuri ja käyttötapaukset pohjautuvat SGEM-projektissa aiemmin saatuihin tuloksiin. Testiympäristön tarkoituksena on toteuttaa osa malliarkkitehtuurista, tämän työn tavoittena on erityisesti integraatiokomponenttina toimivan BizTalk-pohjaisen palveluväylän toteutus. Testiympäristö ei siis sisällä kaikkia malliarkkitehtuurin osia, ja siihen tulee voida myöhemmin lisätä uusia järjestelmiä. Käyttötapaukset toimivat esimerkkeinä, ja uusia käyttötapauksia tulee voida jatkossa testata demonstraatioympäristön avulla.

Testiosuus perustuu käyttötapausten yksityiskohtaiseen analysointiin ja toteutukseen siinä määrin kuin se on testiympäristössä mahdollista. Analysoinnin lähtökohtana perehdyttiin integroitavien järjestelmien välisiin tiedonsiirtotarpeisiin jokaisen eri käyttötapauksissa. Tiedonsiirtoa havainnollistettiin tietovuokaavioiden avulla. Tietovuot ovat hyödyllinen apuväline myös integrointiin liittyvien tietoturvariskien ja -vaatimusten analysoinnissa.

Työn kolmannessa osassa käydään läpi tulokset ja johtopäätökset. Testiympäristöä rakennettaessa ja käyttötapauksia analysoitaessa kävi ilmi, että kokonaisuudessa on vielä suuria puutteita. Testiympäristön integraatiokomponentti eli BizTalk asennettiin ja sillä suoritettiin yksinkertaisia testejä. Käyttötapausten toteutus jäi puutteelliseksi osaltaan siksi, että ympäristön monia muita järjestelmiä ei ollut saatavilla. Kuitenkin jo käyttötapausten analysointivaihe toi ilmi monia ongelmakohtia. Havaitut ongelmat ja niihin liittyvät kehitysehdotukset on käyty läpi käyttötapauskohtaisesti seitsemännessä luvussa.

Kahdeksas luku esittelee käyttötapausten analysoinnista opittuihin asioihin pohjautuvan ohjeistuksen, jota voidaan käyttää tulevien käyttötapausten suunnittelussa. Yhdessä BizTalk-luvun teorian ja asennetun BizTalk-ympäristön kanssa ohjeistus helpottaa ympäristön jatkokehitystä. Ohjeiden mukaisen prosessin avulla uusien käyttötapausten analysointi ja suunnittelu ja sitä kautta tietoturvallisen integraation rakentaminen helpottuu.

Jakeluverkon tietojärjestelmien turvallinen ja toimiva integraatio on älykkään sähköverkon toteutuksen avaintekijöitä. Palveluorientoitunut arkkitehtuuri, palveluväylä sekä yhteinen tietomalli voivat tarjota ratkaisuja integraation haasteisiin. Johtopäätöksenä voidaan kuitenkin todeta, että ne vaativat merkittäviä muutoksia sekä ajatusmalleissa että ohjelmistojen ja integraation toteutustavoissa. Ne eivät ole integraation hopealuoteja eivätkä olemassa olevan arkkitehtuurin päälle liimattavia komponentteja, jotka ratkaisisivat integraatio-ongelmat. Lisäksi erityisesti palveluorientoituneisuus vie pohjan monilta pitkään käytössä olleilta tietoturvan toteutustavoilta ja vaatii uutta ajattelua myös tietoturvaratkaisuihin.

Olennaisen tärkeää on ymmärtää palveluväylän erot perinteisempiin integraatioratkaisuihin nähden ja verrata näitä toteutusvaihtoehtoja integraatiolle asetettuihin vaatimuksiin. Jakeluverkko-operaattorin tietojärjestelmät ovat monoliittisia, eivätkä ne välittömästi muutu palvelupohjaisiksi. Ala kehittyy muutenkin hitaasti muun muassa sähköverkon toiminnan kriittisyyden vuoksi. Lisäksi toimintaympäristö pysyy suhteellisen samanlaisena, vaikka muutokset tulevaisuudessa lienevätkin aiempaa nopeampia. Tällaisessa ympäristössä myös perinteinen, monoliittinen viestinvälityspalvelin saattaa olla hyvä integraatioratkaisu. Integraatioratkaisut kehittyvät kohti palvelupohjaisuutta ja dynaamisen palveluväylän hyödyntämistä, mutta käytännön toteutuksen vaatimat merkittävät muutokset tulee ymmärtää ja huomioida. Tämän työn perusteella ESB-pohjaisen palveluorientoituneen integraatioratkaisun käyttöönotto sähkön jakeluverkkoympäristössä vaatii huomattavaa jatkokehitystä. Työn teoriaosuus toimii johdantona aiheeseen, ja tuloksena kehitetty ohjeellinen prosessi tarjoaa perustan käytännön toteutuksen kehittämiseen.

PREFACE

The work for this Master’s thesis was carried out within the Smart Grids and Energy Markets (SGEM) programme at the Department of Automation Science and Engineering. The program is coordinated and funded by the Cluster for Energy and Environment (CLEEN), a consortium of various companies and research institutions.

I would like to thank my examiner, Professor Hannu Koivisto, and supervisor, Researcher Mikko Salmenperä, for their guidance, comments, and support during this work. I am also grateful to Researcher Jari Seppälä for his comments and advice. I wish to thank my co-workers in the project, and the Department for providing a pleasant working environment and an interesting topic for research. Finally, I would like to thank my family and friends for their continuous support throughout my studies and the writing of this thesis.

The main results of this thesis work will be presented at the 20th annual *Automaatio* seminar of the Finnish Society of Automation, “Automation and Systems without Borders – beyond Future” in May 2013. The co-authors of the seminar paper “Design and analysis of secure integration solution for Smart Grids” include Mikko Salmenperä, Risto Eerola, Jari Seppälä, and Hannu Koivisto.

Tampere, May 20th, 2013

Risto Eerola

CONTENTS

1 Introduction 1
2 Smart Grid 3
2.1 Electrical networks today 3
2.2 Towards a smarter grid 5
2.2.1 Need for a smarter grid 5
2.2.2 Enabling technologies 6
2.2.3 Visions for the future grid 7
2.3 Smart Grid research and development 7
2.4 Information security and real-time aspects 8
3 Integration of information systems 11
3.1 The evolution of integration architectures 11
3.1.1 Hub-and-spoke and Enterprise Application Integration (EAI) 12
3.1.2 Enterprise Service Bus (ESB) 13
3.2 The Canonical Data Model (CDM) 14
3.3 Publish-Subscribe messaging pattern 15
3.4 Service Oriented Architecture (SOA) 16
3.5 Using ESB to solve integration issues 18
3.5.1 ESB is internally service-oriented 19
3.5.2 Dynamic itinerary-based routing 20
3.6 Software solutions for integration 21
3.7 Middleware and SOA performance evaluation 21
3.8 Information security aspects 22
4 Integration landscape: Smart Grid and the distribution domain 25
4.1 DSO information systems 27
4.1.1 Supervisory Control and Data Acquisition (SCADA) 29
4.1.2 Distribution Management System (DMS) 29
4.1.3 Network Information System (NIS) 30
4.1.4 Customer Information System (CIS) 30
4.1.5 Advanced Metering Infrastructure (AMI) 30
4.2 Smart Grid and integration standardisation efforts 31
4.2.1 Roadmaps, frameworks, guidelines, and recommendations 31
4.2.2 IEC: Common Information Model (CIM) 34
4.2.3 W3C recommendations and integration 35
4.2.4 Smart Grid information security standards 36
4.3 Data flows within the distribution domain 37
4.3.1 Data flow diagrams (DFDs) 37
4.3.2 Example DFDs 38
4.4 Using DFDs in information security analysis 39
5 Microsoft Biztalk Server 41
5.1 Introduction to BizTalk 41
5.1.1 BizTalk components on a high level 42

5.1.2 Common enterprise usage of BizTalk 43
5.1.3 Technical point of view 43
5.2 BizTalk key concepts and message flow 44
5.2.1 Receive ports and receive locations 45
5.2.2 Adapters 46
5.2.3 Pipelines and pipeline components 47
5.2.4 Schemas 48
5.2.5 Maps 49
5.2.6 The messaging infrastructure 49
5.2.7 Orchestrations 50
5.2.8 Send ports and send port groups 51
5.2.9 BizTalk databases 52
5.3 The lifecycle of a BizTalk integration solution 52
5.3.1 Installation 52
5.3.2 Solution development and deployment 53
5.3.3 Runtime environment 54
5.3.4 Administration 55
5.4 BizTalk as an Enterprise Service Bus 56
5.5 Information security in BizTalk 60
6 Example architecture and the demonstration environment 61
6.1 Example DSO ICT architecture in SGEM 61
6.1.1 ABB DMS 600 and ABB MicroSCADA 62
6.1.2 OpenEMS Aggregator 62
6.1.3 OpenCIM Calculation Engine 62
6.1.4 Cybersoft Network Manager 63
6.1.5 Other systems 63
6.2 Demonstration environment 64
6.2.1 Virtualisation environment and tools 64
6.2.2 Configuration of the environment 65
6.2.3 BizTalk 2010 environment 66
6.2.4 BizTalk 2013 beta environment 67
7 SGEM Smart Grid use case examples 69
7.1 Case A: Network model exchange 69
7.1.1 Data flows 70
7.1.2 Information security 71
7.1.3 Implementation with BizTalk 73
7.1.4 Lessons learned 74
7.2 Case B: Fault repairing 76
7.2.1 Data flows 76
7.2.2 Information security 78
7.2.3 Implementation with BizTalk 78
7.2.4 Lessons learned 79

7.3 Case C: Active voltage control 80
7.3.1 Data flows 81
7.3.2 Information security 81
7.3.3 Implementation with BizTalk 82
7.3.4 Lessons learned 82
8 The design and implementation of secure integration 83
8.1 Create an essential model through business analysis 83
8.2 Define the use case explicitly 84
8.3 Determine the participating systems 86
8.4 Define the orchestration of the process 86
8.5 Define and implement services 87
8.6 Define data flows to be implemented by middleware 88
8.7 Define the information content of data flows 89
8.8 Define information security requirements 89
8.9 Choose information security implementation methods 90
8.10 Implement the solution 90
8.11 Further considerations: testing, maintenance, and modifications 91
9 General results and discussion 92
9.1 Integration and service-orientation 92
9.2 Information security 94
9.3 BizTalk 95
9.4 Discussion 95
10 Conclusion 98
References 99

ABBREVIATIONS AND TERMS AND DEFINITIONS

AM/FM/GIS	Automated Mapping/Facilities Management/Geographical Information System
AMI	Advanced Metering Infrastructure
AMM	Automated Meter Management
AMR	Automated Meter Reading
AC	Alternating Current
AD	Active Directory
AIN	Automation and Information Networks research group
API	Application Programming Interface
APT	Advanced Persistent Threat
AVR	Automatic Voltage Regulator
B2B	Business-to-Business
BAM	Business Activity Monitoring
BizTalk	Microsoft’s Enterprise integration platform
BPA	Business Process Automation
BPEL	Business Process Execution Language
BRE	Business Rule Engine
CDM	Canonical Data Model
CIA	Confidentiality, Integrity, Availability
CIS	Customer Information System
CIM	Common Information Model
CLEEN	Cluster for Energy and Environment
CVC	Coordinated Voltage Controller
DC	Direct Current
DC	Domain Controller
DER	Distributed Energy Resource
DFD	Data Flow Diagram
DG	Distributed Generation
DHS	Department of Homeland Security
DMS	Distribution Management System
DoS	Denial-of-Service
DR	Demand Response
DSM	Demand Side Management
DSO	Distribution System Operator
EAI	Enterprise Application Integration
EEGI	European Electricity Grid Initiative
EMS	Element Management System
ENISA	European Network and Information Security Agency
ENTSO-E	European Network of Transmission System Operators for Electricity

EPRI	Electric Power Research Institute
ESB	Enterprise Service Bus
ESBT	BizTalk ESB Toolkit
ESX/ESXi	VMware’s bare-metal hypervisor
EV	Electrical Vehicle
FFMS	Field Force Management System
FTP	File Transfer Protocol
GAC	Global Assembly Cache
GIS	Geographic Information System
GWAC	GridWise Architecture Council
HEMS	Home Energy Management System
HTTP	Hypertext Transfer Protocol
ICS	Industrial Control System
ICT	Information and Communication Technologies
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IEEE	Institute of Electrical and Electronics Engineers
IIS	Internet Information Services
IRM	Interface Reference Model
ISA	International Society of Automation
ISO	International Organization for Standardisation
LOB	Line-of-Business
LV	Low Voltage
LVDC	Low Voltage Direct Current
MOM	Message-Oriented Middleware
MS	Microsoft
MV	Medium Voltage
.NET	Microsoft’s software framework
NERC	North American Electric Reliability Corporation
NERC CIP	NERC Critical Infrastructure Protection
NIC	Network Interface Card
NIS	Network Information System
NIST	National Institute of Standards and Technology
NIST IR	NIST Interagency Report
NIST SP	NIST Special Publication
NRECA	National Rural Electric Cooperative Association
OPC	Open Platform Communications (formerly Object Linking and Embedding for Process Control)
OPC DA	OPC Data Access
OS	Operating System
POP	Post Office Protocol

PQ	Power Quality
QoS	Quality of Service
RBAC	Role-based Access Control
RDF	Resource Description Framework
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SDLC	System Development Life Cycle
SGEM	Smart Grids and Energy Markets
SGIP	Smart Grid Interoperability Panel
SGIP CSWG	SGIP Cyber Security Working Group
SHOK	Strategisen huippuosaamisen keskittymä (Strategic Center for Science, Technology and Innovation)
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Simple Query Language
SSO	Single Sign-On
TC	Technical Committee
TCP	Transmission Control Protocol
TMA	Threat Model Analysis
TPE	BizTalk Tracking Profile Editor
TSO	Transmission System Operator
UDDI	Universal Description, Discovery, and Integration
UML	Unified Modelling Language
VM	Virtual Machine
VMM	Virtual Machine Manager
W3C	World Wide Web Consortium
WCF	Windows Communication Foundation
XLANG/s	Microsoft’s “programming in the large” language that BizTalk Orchestrations use to define business processes.
XML	eXtensible Markup Language
XPath	XML Path Language
XSD	XML Schema Definitions
XSLT	eXtensible Stylesheet Language Transformation

1 INTRODUCTION

Constantly available, reliable and affordable electric energy is a crucial element of modern society. The basic technology supplying electricity for everyday needs has served the world for more than a century, and has served it well. However, there is urgent need for major improvements. Without significant upgrades and investment, the ageing electric gird will not be sufficient for the requirements of tomorrow. Demand for energy increases rapidly as the world population continues to grow, countries are developing and standards of living improve. At the same time, the non-renewable energy resources, upon which our energy economy is built, are diminishing with alarming speed.

Designing and building a better, more intelligent electric grid plays a major role in solving these energy issues. Tomorrow's more intelligent, highly automated Smart Grid will support bidirectional flow of both energy and information. It is the key enabler in utilising more sustainable ways of producing energy and more efficient ways of consuming it.

Smart Grids and Energy Markets (SGEM) project studies widely the landscape of future's electric energy solutions. This thesis is part of the project and its main focus is on two important areas within the Smart Grid: integration and information security. The entire Smart Grid is a vast field for research. This thesis concentrates on the operations and solutions of the electricity distribution domain.

In the utilities industries, like in almost any field, information systems are growing both in complexity and in numbers. A common problem is that information and functionality remains locked within isolated systems. Efficient integration of these systems provides many benefits, but is often challenging. The goal of integration is to provide new functionality and new possibilities, as well as increase the efficiency and level of automation of existing processes. While this is a good and desirable thing, new possibilities always go hand in hand with new vulnerabilities and threats. Thus, the integration solution should take information security aspects into consideration. Smart Grid's role as an important part of national critical infrastructure further emphasises the role of information security.

Significant research in both integration and information security has been done throughout the years. The theories are well formulated and often actually quite simple. For example, concepts such as service orientation, loose coupling, authentication, or encryption are clearly advantageous, and on a high level of abstraction, relatively easy to grasp. Yet in practice, integrating or securing systems remains extremely challenging and attempts are not always successful.

Throughout the years, the electricity distribution domain and its ICT architecture have been a focus for research and development. In Tampere University of Technology (TUT) alone, many publications have covered the integration requirements within a distribution network, and offered possible solutions [113;114;137;138;145]. Some of these examples date back to late 1990s, and the topic has been researched even before that. Analysing information security in the Smart Grid from the point of view of home automation is among the recent research topics in the Automation and Information Networks (AIN) research group (where this work was also done) [110]. Clearly, the main problems and needs for improvement have been recognised long ago. The core requirements for the ICT architecture have developed through the years as well. Currently, it is often recommended to develop a Service Oriented Architecture (SOA), which facilitates a modern ESB solution to integrate the systems [81].

After defining a conceptual architecture, the next step is typically to build a prototype or a proof-of-concept solution. As a starting point for this work, example ICT architecture was given, along with three possible use cases that utilise the architecture. Microsoft BizTalk Server was chosen as the integration product for this project.

The objective of this work was to build a demonstration environment to provide concrete results on how an ESB-based integration solution works. The demonstration environment built partially implements the given example architecture. The goal was to provide the ESB component, which can then be used to connect various IT systems and to test different use cases. Thus, this work provides details on both the architecture in general, as well as BizTalk as a specific product. Information security spans through the entire process, and it was given special consideration throughout the work.

The analysis of the use cases started with resolving the data flows between the systems that need to be integrated. These flows were then represented with diagrams, and the information content was analysed. This made it easier to analyse the security requirements for the contained data, and serves as a basis for the integration solution design. It should be possible to integrate systems incrementally, adding one part at a time. Therefore, one goal is to provide some guidelines for a process that will be helpful when adding more systems to the integration and implementing new use cases.

The thesis is organised into three parts. First part (Chapters 2-5) of the thesis describes the landscape and theoretical background. It provides an introduction to the Smart Grid, various integration architectures and their characteristics, the electricity distribution domain and related information systems, most important standards, and Microsoft BizTalk as an example of integration software product. Information security aspects of each topic are discussed. Second part (Chapters 6&7) describes the example ICT architecture and three use cases, which were used as a starting point for experimentation. Third part (Chapters 8-10) describes the results and conclusions. As a result of this work, a BizTalk integration component and a few other parts of the demonstration environment are now installed. This, along with the BizTalk information in Chapter 5, serves as a foundation for future work. The guideline process described in Chapter 8 will help in designing and implementing more use cases in the future.

2 SMART GRID

This chapter explains shortly how the electric grid works today, why it needs to be upgraded, and what enables the Smart Grid. It describes the vision for future development of the Smart Grid, and discusses some research initiatives. Information security considerations specific to the Smart Grid are also discussed.

The globally interconnected electrical networks are suggested to constitute the largest and most complex construction ever built by mankind [42]. Rather than a single entity, it is a system-of-systems. With more and new types of monitoring and controlling capabilities, the Smart Grid will be even more complex. This brief introduction to the Smart Grid includes the basics for readers with little or no background in electrical engineering.

2.1 Electrical networks today

In today's grid, electricity has a typical route from power plants to the consumers. The electrical flows in Figure 2.1 below illustrate the process. The grid itself consists of four main domains: power generation, transmission network, distribution network, and consumer or customer. Additional supporting domains are the network operations, the markets for electricity, and service providers.

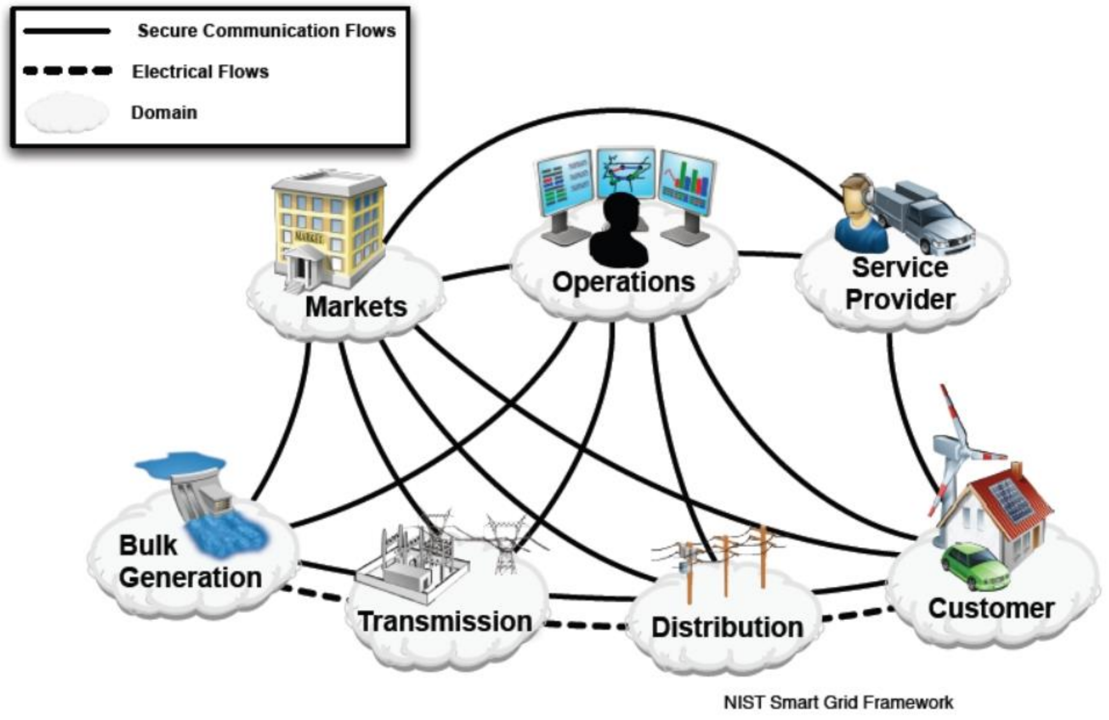


Figure 2.1. *NIST Smart Grid framework conceptual model [102].*

Today, production is concentrated on large, central power plants (such as nuclear, coal, gas, and hydro plants). The transmission network is used to transfer large quantities of power throughout a wide geographical area, using high voltages (from 110 kV upwards). The network can cover the entire country, and the transmission system operator (TSO) is usually owned and/or controlled by the state. Nationwide transmission networks may be connected to other countries' networks, as is the case, for example, in the Nordic countries.

The medium-to-low-voltage (110-20-0.4 kV) distribution networks operate in smaller geographical areas and distribute electricity from the transmission network to the customers. A distribution system operator (DSO) owns and operates the distribution network within a certain area. DSOs are local monopolies, as the networks are very capital-intensive investments, and building multiple networks within a single area would not make any sense. To avoid abuse of the monopoly position and ensure reliable operations, the distribution business is usually strictly regulated.

The entire electrical network together with its operation and the supporting markets comprises a vast system of interconnected subsystems. Governments and other regulating bodies, as well as standardisation organisations, have an influence on the development of the grid. Additional stakeholders are, for example, the companies that manufacture the various products for building the network and its supporting systems. For the purposes of this thesis, the generation and transmission domains, as well as the customer point of view, are of less interest. The focus is on various information systems used in the distribution domain and its supporting operations.

The way electricity is produced, transmitted, and distributed today has many drawbacks. For example, there are no cost-effective solutions for storing large amounts of electricity. As a result, production and consumption (including line losses) must be in balance at any given time. Today the network is operated so that production follows consumption, meaning that production is adjusted as consumption varies. This is especially challenging in the distribution domain: the load pattern varies dynamically with time, it is hard to predict, and cannot be adjusted [38, p.142].

Backup production capacity is needed in order to meet peaks in demand. This backup capacity is, however, poorly utilised; its use may total just a few days per year. Keeping the capacity in place is expensive because it yields returns for the investment only when used. In addition, the passive network (wires and components) has to be designed and built with excess capacity to withstand the peak loads.

In transmission networks the remote monitoring and control capabilities are relatively high [42, p.513]. In distribution networks the structure is more complex, and the degree of automation is much lower. The medium-voltage (MV) feeders feature limited remote control capabilities. In the case of low voltage (LV) networks, the operator is essentially blind; there is no sensory data of the status of the network. The operators cannot see the grid, and even if they could, without available control systems there is no way to react [73]. For example, while LV networks are fuse-protected, fault location is based purely on customer reports.

2.2 Towards a smarter grid

The basic technology of the electric grid dates back to the 19th century. The oldest installed parts still in operation could technically be from that time. In many countries, large portions of the network are approaching the end of their lifecycle. Major parts of the distribution network in Finland were built decades ago. The existing network is outdated and needs to be improved, both in the sense of technological ideas as well as the concrete installation. [51, pp.4-5] The long lifecycle (often many decades) implies two things. First, the average age of the existing installation is rather high. Second, the updates done today may well be in place for a half of a century. Thus, appropriate decisions in planning and implementation are crucial.

The rapid development of ICT over the past decades provides means for significant improvements in the grid. The ageing assets have to be replaced in any case, which makes this the perfect time to upgrade to the Smart Grid.

2.2.1 Need for a smarter grid

Largely based on fossil fuels, our current energy systems are evidently not environmentally sustainable. Renewable energy sources have the potential to provide plenty of clean energy. In the future, extensive distributed generation (DG) is required in addition to large central power plants. For details on distributed generation, see, e.g., [6;136].

However, the output of renewable energy and DG installations is unpredictable and fluctuates in response to natural conditions. The traditional, passively managed distribution grid would require additional backup power generation resources and massive investments in wires and equipment in order to accommodate to the fluctuation. This would further decrease the utilisation rate of the network – in times when economic reasons call for increased rate. Another option is a more intelligent, actively managed grid. Demand response (DR) and demand side management (DSM) allow for the intelligent adjustment of consumption to the currently available level of production.

There are a few ways to achieve this. Functions such as space or water heating, or operation of a washing machine, can take place in times of non-peak load, without affecting the consumer’s life significantly. This will help reduce the peak load. Alternatively, these activities can be performed when there is excessive production (from renewable sources). With near real-time pricing information, smart appliances can be programmed to switch themselves on or off depending on the price of electricity.

As electrical vehicles (EVs) become more popular, their batteries will offer possibilities for large-scale distributed energy storage. This is another example of DR/DSM. The batteries can be charged when there is surplus production due to, e.g., strong wind conditions. During peak load times, some energy can then be drawn from the batteries, which lowers the need for backup generation capacity.

Increased consumer awareness is also a desired outcome. Providing consumers with more information and more and better ways to manage consumption will hopefully lead to energy savings. Individually, the effects may be small, but when combined they can make a large difference.

Environmental concerns and growing energy demand are not the only reasons to upgrade the grid. The modern society is highly dependent on electricity. Securing the supply with proper infrastructure is a priority task for governments around the world. The grid needs to be secured against malicious attacks as well as natural phenomena. Extreme weather conditions due to climate change are increasingly probable, and the adverse effects of losing electricity supply are more severe, as so many things depend on electricity. Further, many high-technology devices such as computers have higher requirements in terms of the quality of electricity, demanding more from the grid.

All these scenarios call for a smarter grid that offers bidirectional flow of both energy and information. Better customer service, improved market for electricity, as well as overall reliability and security are also important drivers and needs related to the Smart Grid. A list of drivers and needs of the Smart Grid is given in [75].

2.2.2 Enabling technologies

Various technological improvements and innovations will enable the envisioned future. The development of ICT in the recent years is one of the main reasons why a smarter grid is now an actual possibility. The key is to provide more information to base decisions on (measurements) and better decision-making solutions (controls). Keeping the costs affordable and providing information in a real-time manner are major challenges. Yet in most cases the technology exists; it is about applying it successfully - in a scale never seen before.

The future development of increased intelligence (that is, penetration of ICT) in the grid is illustrated in Figure 2.2. The equilibrium point will move to the right on the horizontal axis as both of the curves shift. The cost curve will shift downward along with cheaper technologies and the value curve upwards as the more intelligent grid will provide new usage scenarios and benefits.

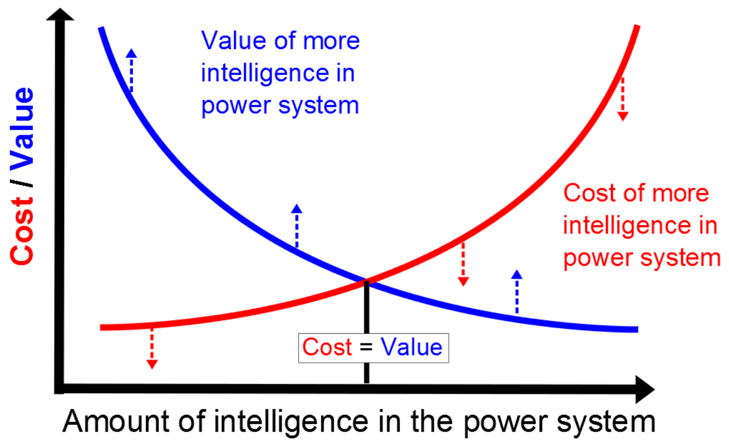

Cost / Value
Value of more intelligence in power system
Cost of more intelligence in power system
Cost = Value
Amount of intelligence in the power system

Figure 2.2. *Amount of intelligence in the grid, adapted from [134].*

From the customer point of view, the most prominent and obvious development is the introduction of the smart meter. It is a key enabler of a smarter grid, acting as a customers' gateway to the grid. From the utility point of view, however, it is only one improvement among others. The smart meter also involves certain problematic issues and threats, especially in the fields of information security and privacy.

The development of electric vehicles (EVs) is important for the whole Smart Grid. Key issues in EVs are the battery capacity and recharge speed. Widespread use of EVs will pose challenges to the gird. An infrastructure of charging stations and outlets needs to be built. Charging requires intelligence as well; if badly coordinated, it might quickly over-strain the grid [117].

The Smart Grid is an umbrella term covering countless concepts, ideas, and technologies. Promising topics of research include, for example, grid-scale battery storage [82] and superconductivity [80]. Other important aspects not discussed in this short introduction include low-voltage direct current (LVDC) networks, improved power electronics, virtual power plants, power cells, micro grids, and super grids. A more comprehensive listing along with examples is offered in, e.g., [42, pp.508-511].

2.2.3 Visions for the future grid

Building the Smart Grid is a massive effort that will span over the coming decades. High-level visions for the long-term development play an important role in such vast projects. The visions for the Smart Grid are numerous, and there is no single definition for it either. Key differences of the traditional grid and the visions of the future Smart Grid are presented in Table A.1 (Appendix A). These qualities are commonly listed in literature and largely accepted as important aspects of the Smart Grid.

The European Technology Platform for Smart Grids defines the Smart Grid as "an electricity network that can intelligently integrate the actions of all users connected to it - generators, consumers and those that do both - in order to efficiently deliver sustainable, economic and secure electricity supplies" [128]. According to an unknown source, quoted in [74] the Smart Grid is "an attempt to maximize the utilization degree of electricity networks and electricity production capacity by leveraging the latest information technology, two-way communication and system intelligence."

2.3 Smart Grid research and development

Smart Grid is currently a trending topic and subject of interest and major research all over the globe. Or, as the grid modernisation efforts were more bluntly described in [84]: "-- bringing intelligence into this venerable relic of nineteenth-century technology is a worldwide priority." Major players, such as the European Union, the USA, China, Japan, South Korea, and Australia have all started their Smart Grid development and are investing heavily into research in this field. Pilot projects of various scales have also been launched in the recent years, in order to provide concrete results.

Finland has also launched its own Smart Grid development programme. In many ways Finland's grid is already quite advanced, sometimes called "Smart Grid version 1.0" [43;74]. This thesis is done as part of the Smart Grids and Energy Markets (SGEM) programme, which was launched in 2009 under the CLEEN (Cluster for Energy and Environment) Strategic Centre for Science, Technology and Innovation (SHOK, Strategisen huippuosaamisen keskittymä). The programme aims to create a vision and develop practical solutions for the next generation smart grids. For more information, see e.g., [27;29;135]. Public deliverables are available at [28].

2.4 Information security and real-time aspects

In the heart of the Smart Grid is ICT, which introduces countless benefits but also completely new issues of information security. The Smart Grid is not a traditional IT environment. It has special properties that make it an exceptionally challenging and important environment for information security.

The fact that Smart Grid is part of society’s critical infrastructure makes its information security a critical aspect as well. The Smart Grid can even be considered more critical than most other parts of the infrastructure, as so many things depend on electricity. Critical infrastructure is a prime target for advanced attacks, performed by adversaries with utmost capabilities and resources (e.g. Advanced Persistent Threats [APTs], or full-blown cyber warfare between nation-states). This must be taken into account in the design of Smart Grid information security - even though it might be impossible to be completely safe from such attacks.

The Smart Grid is a combination of traditional IT and automation systems. Here, automation refers to Industrial Control Systems (ICSs), as industrial automation traditionally has its role in electricity generation, transmission, and distribution. However, in the Smart Grid vision, the customer is no longer a passive consumer of electricity, and aspects of home automation will be increasingly important. Information security of home automation is an important issue, but will not be discussed here.

Experts in fields of IT and automation look at security from a very different point of view [30]. While automation systems share some basics with IT systems, they are technically, administratively, and functionally more complex and unique [147]. Yet when combined, usually the smaller control network “joins” the larger, more mature enterprise IT network [30]. An obvious difference in the nature of the systems is that automation monitors and controls the physical realm around us. Breaches in its information security can potentially have very concrete, direct consequences. [147]

Using up-to-date software is a crucial information security method. This is challenging in the automation industry, which is notoriously slow to adapt to change. Each change poses a threat to the continuous operation of the process, and thus must go through a rigorous and time-consuming testing process before acceptance. Automation systems, and many parts of the Smart Grid, have a lifespan of decades rather than years. Further, these systems may operate continuously for months, with no possibility for software updates or restarts. In general, the information security of ICSs is said to be up to a decade behind the enterprise IT [116]. Thus, compared to IT, automation systems will use older technology that can neither be replaced very often, nor updated rapidly. For more details on ICS information security, see e.g. [8;78;131;133].

The traditional triad of IT security, (confidentiality, integrity, and availability, or C-I-A) is applicable to ICSs, but the priority is reversed (A-I-C) [147]. Here, availability is used in a very general sense: data needs to be available to the intended users, and within a specified timeframe. In this typical ICT definition, the aspect of time is added almost as an afterthought. Arguably, on a high level of abstraction, the time aspect of availability is a requirement for control systems as well (a controller needs to have measurement data available, and at a specified time). However, the term "availability" alone is insufficient, and, in fact, hardly ever used. Discussion of automation information security must include more detailed definition of real-time requirements.

The following examples will clarify how the conception of "sufficient availability" is very different for ICT and ICS realms. For example, resending lost data is a common method in communication protocols: if a sent Transmission Control Protocol (TCP) packet is not acknowledged as received, sender will try resending it. The data is still considered available, if it reaches the destination after resending. If a hard drive fails, but a recent backup can be restored, the data is considered available. If a website is unavailable for a short while, but then can be reached again, it could still be considered available according to its Service Level Agreement, SLA (e.g., 99.99% availability). Having data available in a sense that it is never lost is important for IT systems. In control systems, data that is not there at the exact moment it is needed is generally bad data; it is useless and could lead to erroneous operation and system failure [35, p.3].

The concept of utility (how the utility or usefulness of the information changes as a function of time), is helpful when discussing the timeliness issues [79] (Figure 2.3).

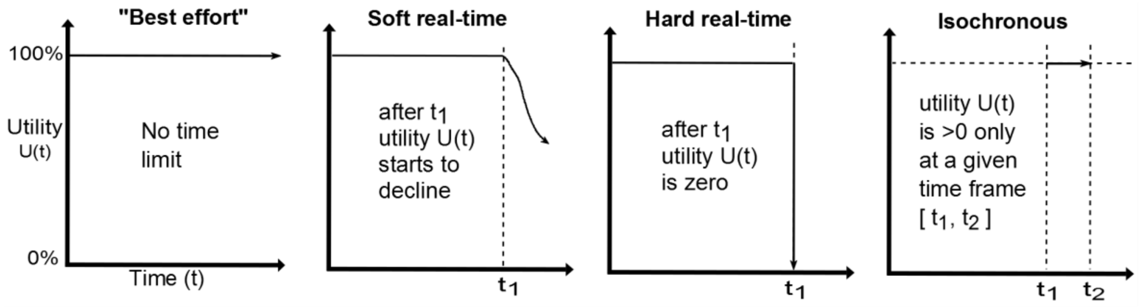

"Best effort"
100%
Utility U(t)
No time limit
0%
Time (t)
Soft real-time
after t_1 utility U(t) starts to decline
t_1
Hard real-time
after t_1 utility U(t) is zero
t_1
Isochronous
utility U(t) is >0 only at a given time frame [t_1, t_2]
t_1
t_2

Figure 2.3. *Concept of utility and types of real-time requirements, adapted from [79].*

In best effort operation, there is no deadline; utility does not change over time. Hard real-time requirement means that data must be available before the deadline, without exception. Soft real-time requirements are less demanding, and can either be missed sometimes, be missed with small time deviations, or occasionally even ignored. [35, p.3] Isochronous means that data is only useful within a specific time frame.

For many ICT solutions, best-effort operations are sufficient. However, real-time systems do not operate correctly if the timeliness, performance, and schedulability requirements cannot be met [35]. This is a major concern in automation systems, and designing and building real-time systems and software is an art of its own (for further information, see, e.g., [35]).

In control systems, these are traditionally assessed mainly as safety considerations, which will not be discussed here. However, information security, which is the point of view in this work, does contribute to the overall safety, thus the issues overlap. Intentionally affecting real-time performance through means of network attacks, for example, is an information security issue which could affect the overall safety of the system. A cyber-attack against automation systems can cause major problems with mere addition of transmission delay into the control network.

Any modern automation system is increasingly an ICS/ICT system combination, thus information security is a valid concern. However, what differentiates the Smart Grid from any earlier system is the staggering size. The complexity of the current electrical network (let alone the Smart Grid) is a threat in itself. Tightly coupled interconnected mega-systems, such as the Smart Grid, are more efficient, but also more vulnerable [17, see 14]. One potential risk is the uncontrollable and unpredictable propagation of disturbances. Even a relatively small fault, unintentional or malicious, can have major cascading effects [17;146]. Examples of this are the massive blackouts in recent years (e.g. in the USA and India). The network is vulnerable even without any hostile actions.

Cloud computing is an emerging trend, that will likely have many uses in the Smart Grid. It has even been argued that it is the only technology capable of providing the computing power required by the Smart Grid. For example, smart meters will allow measuring intervals to be hourly instead of yearly or monthly, increasing data amounts manifold. Cloud computing promises nearly unlimited computing capacity, but its performance currently falls short in other areas for Smart Grid use (e.g., real-time capabilities, consistency, security, and privacy). [15]

Clearly, the importance of information security in the Smart Grid has been recognised at an early stage. Smart Grid is, to a large degree, a new system, and it is crucial to “build security into it”, rather than try to add it as an afterthought. System development life cycles (SDLCs) indicate that the former approach is highly advantageous. Much has been learned by securing traditional IT systems; the experience should be used to lower the learning curve for Smart Grid security. For a more detailed review on Smart Grid security aspects, see for example [51], and Chapter 4.2.4.

3 INTEGRATION OF INFORMATION SYSTEMS

Different information systems and software are an important part of the Smart Grid. Fundamentally, IT systems offer 1) a way to store and access information, and 2) various functionalities to process that information. In the early days of information technology, software systems operated as isolated containers, without any kind of integration. The number and complexity of IT systems has proliferated, leading to issues of redundancy and inconsistency: the same functionality is implemented in multiple places, and copies of information are stored within various systems.

These times of isolation are now history. As more complex functionality is demanded from the IT systems, the benefits of inter-system communication and integration have become evident. Integration aims to make reuse of the existing functionality simpler, thus helping to remove redundant functionality. It also helps with data inconsistency and redundancy issues, when each system no longer needs to keep its own copy of information. Successful integration increases efficiency and reduces costs and errors.

The idea of systems communicating with each other is seemingly simple but implementation is often far from it. Many challenges arise within heterogeneous IT environments (e.g., incompatibility of data formats, system metadata, wire formats, and message exchange protocols, as well as weak process visibility [122, p.66]). Integration efforts can lead to what Chappell fittingly refers to as “accidental architecture” [23].

Integration, just as the Smart Grid, is a vast and complicated topic. A basic introduction is offered here, with the emphasis on those architectures, patterns, ideas, and technologies that are relevant for this project. A good source on integration is [58].

3.1 The evolution of integration architectures

In today’s connected and networked environment, no software is an island. Looking back, the evolution of integration solutions has advanced in logical steps. When the need to connect two separate systems first arises, the logical thing to do is to directly link the systems together. New links are built as new systems need to be connected.

This sort of ad hoc point-to-point integration became popular, mainly because the design is simple and the implementation straightforward. A sample structure is shown in Figure 3.1. Point-to-point architectures are still used and they work well for a small number of nodes. Also, for some performance-critical applications it might be the best (or even the only) option, as the direct links can be implemented with very little overhead.

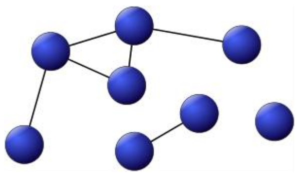


Figure 3.1. *Simple point-to-point integration.*

However, as the number of systems grows, the downsides manifest. One major issue is scalability. When *n* systems need to be connected with each other, the number of unidirectional connections required is *n* * (*n*-1). The number of required connections grows with the square of the number of nodes. This exponential growth quickly leads to a complicated structure, as shown in Figure 3.2, and adding more nodes becomes burdensome. It is hard to monitor such a system as there is no central connectivity point.

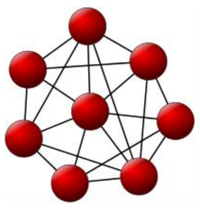


Figure 3.2. *Complex point-to-point integration.*

This sort of integration becomes impossible to maintain because the systems are tightly coupled. The links between the nodes are based on sort of technical contracts that define the connectivity details, such as endpoint location. Changing a node in a way that changes the contract (e.g., updating a system) breaks the integration, and all the links must be updated accordingly. With anything but the simplest cases, the cost of implementing and updating this sort of integration becomes prohibitive. Arguably, the point-to-point architecture could be described more accurately as lack of architecture.

3.1.1 Hub-and-spoke and Enterprise Application Integration (EAI)

The logical next phase in the integration evolution is to add a hub as a central node, to which all the other nodes connect to. This is known as the hub-and-spoke architecture (Figure 3.3), and the corresponding integration of systems is called Enterprise Application Integration (EAI).

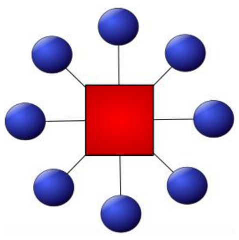


Figure 3.3. *Hub-and-spoke architecture.*

When using the hub, the number of required connections equals the number of nodes, so the growth is linear instead of exponential. This makes the solution significantly more scalable. The active hub can act as a message broker that decouples the senders from the receivers. The endpoints are now loosely coupled. Maintenance is easier because a change in one system only changes the connection between that system and the hub. The hub-and-spoke architecture also enables central monitoring and administration, logging, and traffic flow control. These are hard, if not impossible, to implement in point-to-point solutions.

This architecture has its downsides, too. Each message now has to make two hops instead of one, which makes the path more complex and increases latency. The hub can also become a performance bottleneck as all the messages travel through it. In this sense, pure hub-and-spoke does not scale well. Further, the hub introduces a single point of failure. These issues can be mitigated with the so-called federated architecture, where redundant, interconnected hubs provide load sharing and improve fault tolerance. [109]

The hub quickly grows into a complex structure that is difficult to maintain and expand (even more so if the architecture is federated). All in all, traditional EAI solutions have been criticised for being expensive, monolithic structures based on proprietary technologies, where the hub needs to "know everything and do everything" [37, p.647].

3.1.2 Enterprise Service Bus (ESB)

The Enterprise Service Bus (ESB) emerged to address the shortcomings common to EAI solutions. The bus architecture (Figure 3.4) is seemingly similar to the hub-and-spoke architecture. It seems that, instead of a hub, the central node is just pictured as a bus and renamed accordingly. However, there is more to it than just a new name [24].

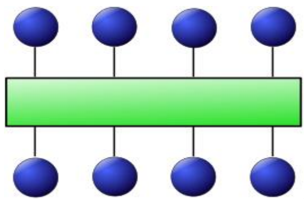


Figure 3.4. *Bus architecture.*

An ESB shares some of the downsides of EAI, and has all the same benefits. Yet, in order to be useful, ESB has to have some additional advantage over EAI. While both architectures separate the application and integration logic, they are differentiated by the distributed nature of the ESB, as shown in Figure 3.5. This, among other differences, will be explained in more detail in Chapter 3.5. As opposed to EAI and ESB, application server and Message-oriented Middleware (MOM) are approaches that have the integration and application logic intertwined.

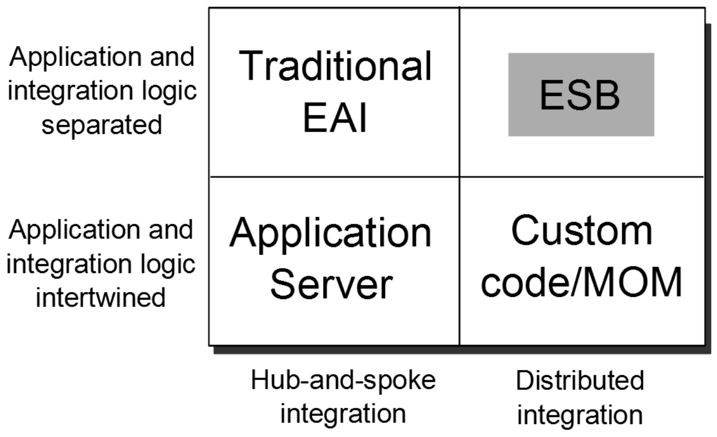

Application and integration logic separated
Application and integration logic intertwined
Traditional EAI
ESB
Application Server
Custom code/MOM
Hub-and-spoke integration
Distributed integration

Figure 3.5. *Different integration approaches, adapted from [23].*

Regardless of the choice of architecture, a poor implementation can spoil a good design: it is possible to build a good point-to-point solution, as well as a bad hub-based solution [26]. Further, the architecture will only address a small subset of the problems common to integration. Other problems and some suggested solutions are explained in the following subchapters.

3.2 The Canonical Data Model (CDM)

The central hub decoupled the sender and receiver in terms of location, and helped to solve the problem of exponentially growing number of physical connections. However, similar problems arise in the data format level. Systems have different ways to represent data internally, and translations from one format to another are required (Figure 3.6).

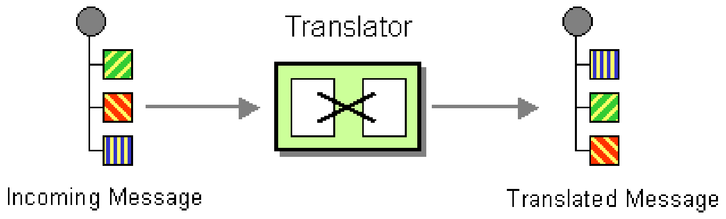


Figure 3.6. *Message translator [60].*

Each system can use a format of its own, wherefore transformations from any format to any other format are required, as illustrated by the green dots in Figure 3.7. Thus, the number of transformations grows exponentially when new systems are added, which again leads to major scalability issues.

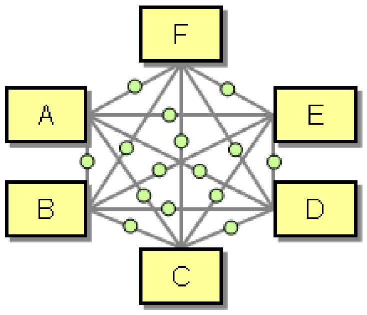


Figure 3.7. *Data format translations [59].*

For a familiar problem, there is also a familiar solution. The concept of a central hub is applicable for many situations [26]. To solve data format issues, it is applied on a metadata level. The resulting “metadata hub”, illustrated in Figure 3.8, is not a physical component; it is a Canonical Data Model, CDM (sometimes Common Data Model).

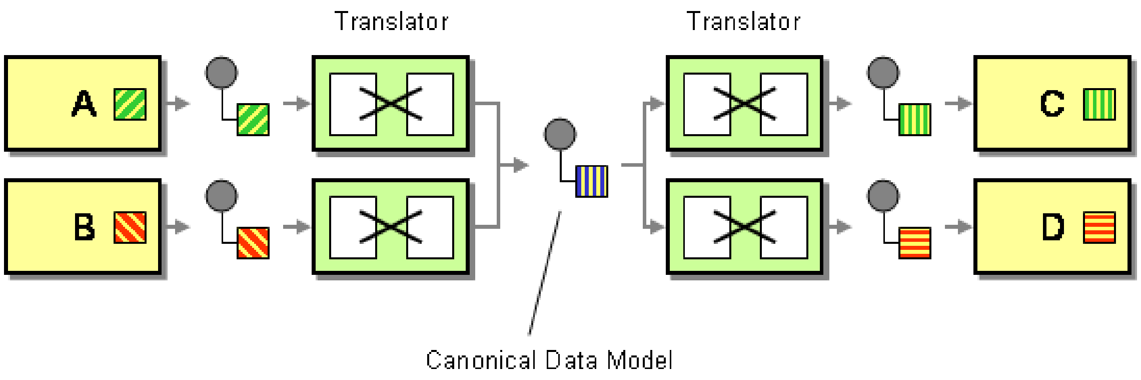


Figure 3.8. *Canonical Data Model [57].*

The canonical format is, in a way, common for all the participants, yet it is independent from all of them. All translations happen between the canonical format and a system-specific format. This effectively decouples the endpoints on a data format level. The number of translations is now equal to the number of nodes: adding a new node means just adding a translation from the system’s specific format to the canonical format. Thus, the growth is again linear and scalability significantly improved.

The hub was again a successful solution to the problems that arise from exponential growth. Unsurprisingly, the same solution will lead to similar new issues. Using a CDM, each passing message needs to be translated twice instead of once. The solution is more complex and latency increases as more computing is required. The CDM must offer a representation for any sort of data contained in the endpoint systems. Just like the monolithic hub, the model can quickly become complex and difficult to comprehend. For additional information, see, e.g., [20, p.397;57;102, p.57].

3.3 Publish-Subscribe messaging pattern

Participants in a message passing system can connect and communicate with each other in various ways. Messaging patterns are one way to describe the communication paradigms. Publish-Subscribe (often: pub-sub, pub/sub), is a messaging pattern that fits well into the hub-and-spoke architecture. The hub acts as a subscription manager, where subscribers register their interest in certain messages. Publishing means simply sending messages to the hub.

With the use of hub-and-spoke architecture and Publish-Subscribe messaging, the integration solution becomes loosely coupled in terms of location, time and synchronisation. A publisher is unaware of how many subscribers there are, where are they located, and in what state they possibly are (e.g., offline or online). Further, the systems are not synchronised: a subscriber does not have to block its execution while waiting for a message or a response. [45] This decoupling increases scalability.

Subscribers usually receive only a small subset of all the messages. Filtering can be based on message topic or content. Topic-based is rather static and primitive as it is the publishers’ responsibility to know the right topic to publish into. Content-based filtering is more dynamic, as messages are classified based on their properties, not some predefined external criteria. The subscribers are responsible of defining what type of messages they wish to receive. This can be highly expressive, but as a downside it requires sophisticated protocols that have higher runtime overhead. For details on pub-sub, see, e.g., [45;62;121, pp.17-19].

3.4 Service Oriented Architecture (SOA)

A move from a component-based towards a service-based architecture was another attempt to avoid the problems of tightly coupled point-to-point integration [50, p.4]. Fundamentally, Service Oriented Architecture (SOA) is an architectural pattern and a set of design principles; it is one approach to organise enterprise IT resources. The key goals of SOA are flexibility, agility and reusability. The underlying idea is that IT and software systems should support, not restrain, business needs. The concept of service-orientation and recognition of its benefits predate the buzzword ‘SOA’ [85;99]. A short introduction is offered here, for details see, e.g., [20;21].

The basic building block of SOA is a service, which, by a general definition, is performance of work by one for another [72]. In SOA, however, the definition is not generic [10]. Using a service is called consuming, and, rather than human end users, consumers are most often other systems, applications, or services.

A service represents a discrete chunk of functionality, described in a published contract which the service adheres to. Beyond this contract, a service is abstract and autonomous, i.e., it encapsulates (hides) the implementation logic and has control over it. Services are loosely coupled, having minimal outside dependencies. Services are stateless, thus improving SOA scalability as state management can be resource-intensive. They are technology-agnostic and context-independent, meaning the technological details of the environment of both consumer and provider and the previous action of caller before service invocation, are irrelevant. Other key qualities include discoverability and accessibility (over a network), and ability to effectively compose complex solutions using multiple services. Last but not least, services are reusable and provide some valuable business functionality to one or, preferably, many consumers. [9;34;44;76;111]

Figure 3.9 shows the three core principles, namely service contract, loose coupling, and abstraction, and their influence on the other principles as described in [44].

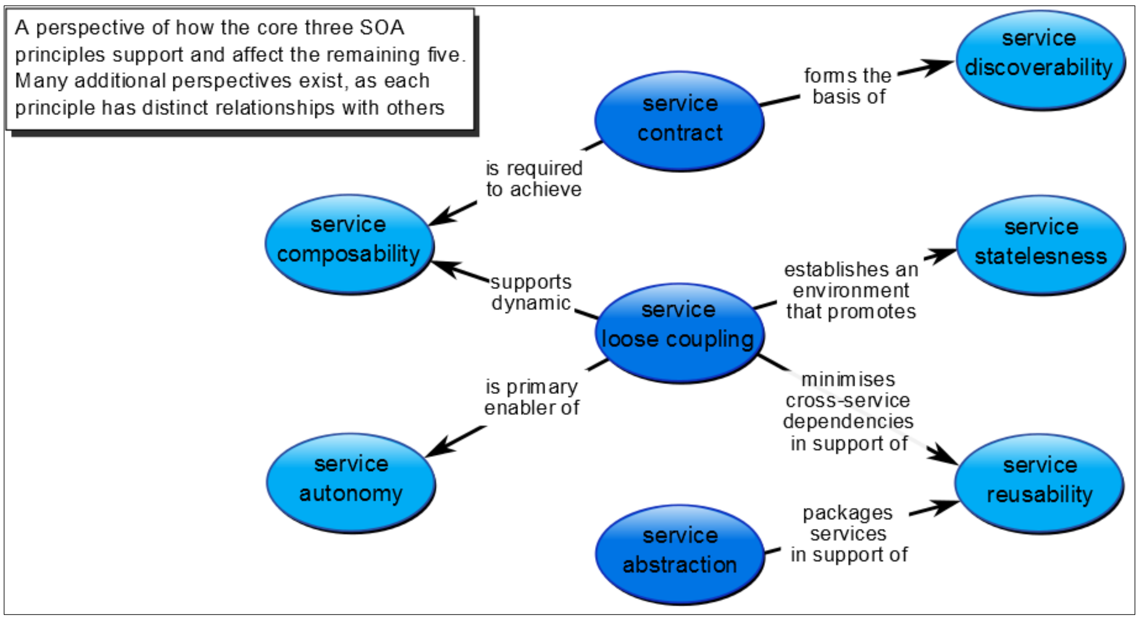

A perspective of how the core three SOA principles support and affect the remaining five. Many additional perspectives exist, as each principle has distinct relationships with others
service contract
forms the basis of
service discoverability
is required to achieve
service composability
supports dynamic
service loose coupling
establishes an environment that promotes
service statelessness
is primary enabler of
service autonomy
minimises cross-service dependencies in support of
service reusability
service abstraction
packages services in support of

Figure 3.9. *SOA principles interrelations, adapted from [44].*

In order to move towards service-orientation, the currently used architecture needs to be broken down into its functional primitives. The information and behaviours (functionality) of the system must be understood. The service-oriented architecture is then built with service interfaces that are abstracted into a configuration layer which is used to create (and re-create) business solutions. [85] This supports the idea that SOA is about architecture rather than application development. More important than implementation of a particular service, is the decision of which services will be created. [99]

Services are rarely created from scratch. It is common to expose functionalities within existing systems as services (i.e., use service wrappers). One key design question is: when or where does it make sense to use services? Not all functionality should be provided as a service, because services introduce certain overhead, both in design work and runtime execution. [21]

Service invocation involves usually three roles and three operations (Figure 3.10).

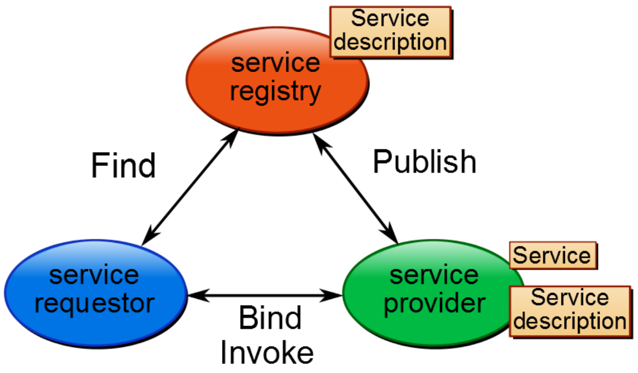


Figure 3.10. *Service invocation roles, operations and artefacts.*

A service provider publishes the service contract in the service registry. By querying the registry, a service requestor will find what it needs. After finding the proper service and obtaining binding information, the requestor binds and invokes (executes) the service.

The entire process is referred to as "find-bind-invoke". The registry is an optional component, as the binding can be direct and static between the provider and requestor, or the information about the service can be obtained through other means (these approaches have obvious downsides). [23, pp.126-127;111, pp. 22-26]

High expectations were placed on the concept of SOA, up to the point of over-hype and eventual disillusionment. Advantages of SOA are clear, but again, a good concept can be spoiled with poor implementation. A haphazard, ad hoc approach to building services can lead to a similar architecture and similar issues as point-to-point systems integration. This sort of uncontrolled "service spaghetti" is illustrated in Figure 3.11.

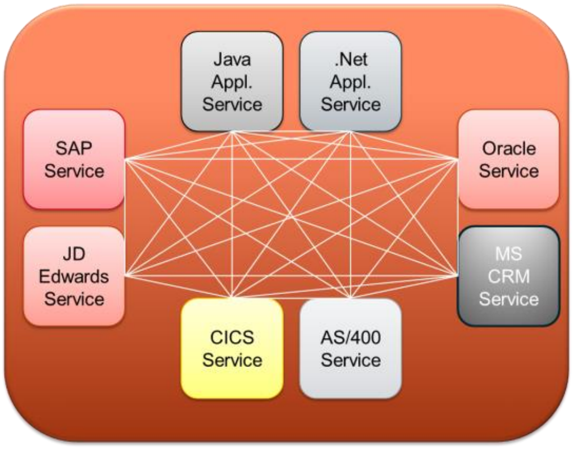


Figure 3.11. *"Service spaghetti" [50].*

Service-oriented applications can end up delivered as a set of point-to-point solutions and become just as tightly coupled as a monolithic application. This kind of implementation faces all too familiar issues (e.g., it does not scale well, has redundant functionalities, is inflexible, and difficult to monitor). In other words, the system loses most (or all) of the benefits of service-orientation and might end up worse off. [50]

3.5 Using ESB to solve integration issues

In EAI solutions, the monolithic nature of the hub became a problem. The move towards SOA held many promises, but many of them were left unfulfilled (at least with the "first wave" of implementations). ESB aims to solve issues discovered in earlier attempts of integration and to serve as a framework for building service-oriented applications. The ESB is often described as one layer, the messaging backbone, of an overall SOA. The idea is to move the logic away from individual endpoints into a logically centralised, loosely coupled, dynamic layer that manages interactions. All the services will connect through a mediation layer provided by the ESB, which helps to solve many point-to-point service connectivity problems. [50]

This idea of a central layer seems very similar to the EAI hub. However, ESB is different from EAI in two important ways. First, the ESB is internally service-based; it offers integration service components that can be distributed across the bus. Second, it is more dynamic in nature. It can resolve certain things (e.g., endpoint location, transformations) runtime, instead of configuring them beforehand at design-time. [24]

3.5.1 ESB is internally service-oriented

In order to improve the monolithic hub architecture, the ESB design is distributed and internally service-oriented. Figure 3.12 illustrates the integration architecture development, showing how the ESB is composed of services that allow distributed deployment.

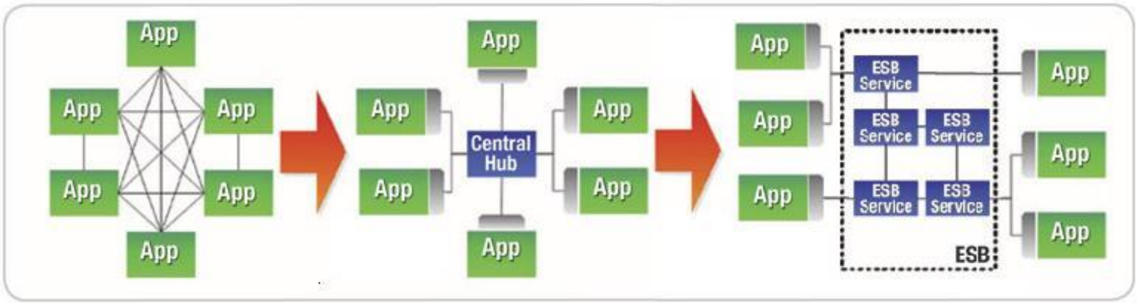


Figure 3.12. *Integration evolution: an ESB is internally service-based [1].*

The base functions of the integration broker are divided up into their constituent parts (i.e. services). These services can be deployed separately and independently across the bus. [12] What the ESB offers is a messaging fabric and a common set of integration components, on top of which developers can build their own services [87, p.640]. Examples of commonly needed functionality that the ESB can offer as services are data transformation, protocol conversion, location and version transparency, and error handling, as shown in Figure 3.13 [50].

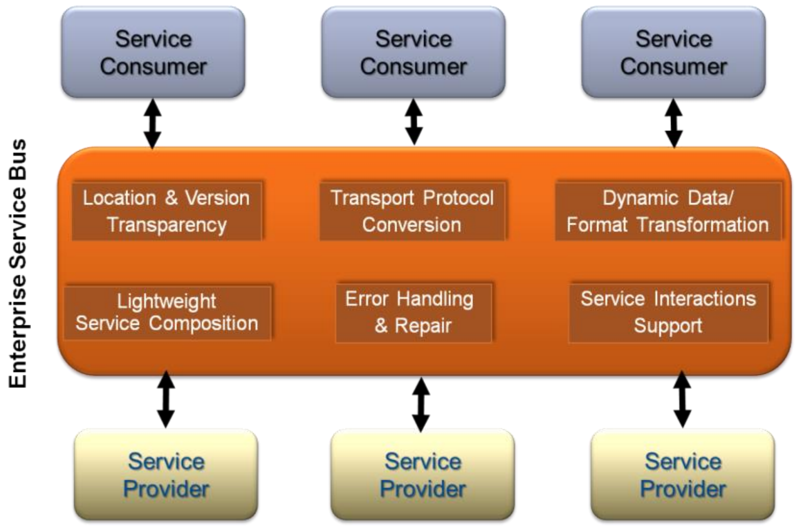

Service Consumer
Service Consumer
Service Consumer
Enterprise Service Bus
Location & Version Transparency
Transport Protocol Conversion
Dynamic Data/ Format Transformation
Lightweight Service Composition
Error Handling & Repair
Service Interactions Support
Service Provider
Service Provider
Service Provider

Figure 3.13. *ESB offers the integration functionality as services [50].*

This architecture is a clear step forward from a single monolithic stack, and it solves some EAI problems. The ESB has no single point of failure. The architecture scales well, as the integration broker functionalities can be deployed selectively, exactly where and when they are needed and without any over-bloating. Co-operating harmoniously, together the services provide all the required integration functionalities. [12;24]

3.5.2 Dynamic itinerary-based routing

In addition to the distributed nature, the emphasis on dynamic execution differentiates the ESB from an EAI solution. The often heard ESB mantra is "configuration rather than coding". The idea is to remove the need for design-time specification and hardcoding of the relationships between interconnected applications. [24] For example, using the find-bind-invoke approach in SOA, each client (service) needs to have code that implements the lookup and invoking [23, p.126]. Changes in code require recompiling and redeployment. Configuration, on the other hand, happens post-deployment.

The core of the problem is: how to route a message through a series of steps (services), when these steps are not known design-time, and may vary for each message? [63]. Content-based routing, offered by many EAI solutions, is a partial solution. It is dynamic in a sense that the endpoints are determined runtime, based on the content of the message. Additionally, the router can refer to a central, configurable rules engine to determine the endpoints. That is clearly dynamic, configuring rather than coding, so how does the ESB improve that?

The problem becomes evident (and is all too familiar) when a message has to be routed through multiple steps. In EAI, the routing logic is implemented in the central hub or rules engine. After each step, it is necessary to refer back to it for instructions. This is again a possible bottleneck and a single point of failure. Itinerary-based routing is offered as a solution. The idea is to attach an itinerary (routing slip) to each message, specifying the sequence of the processing steps (Figure 3.14). Technically, the message could already have an itinerary attached to it once it enters the bus, but in most scenarios, the ESB will dynamically resolve the correct itinerary and attach it to the message.

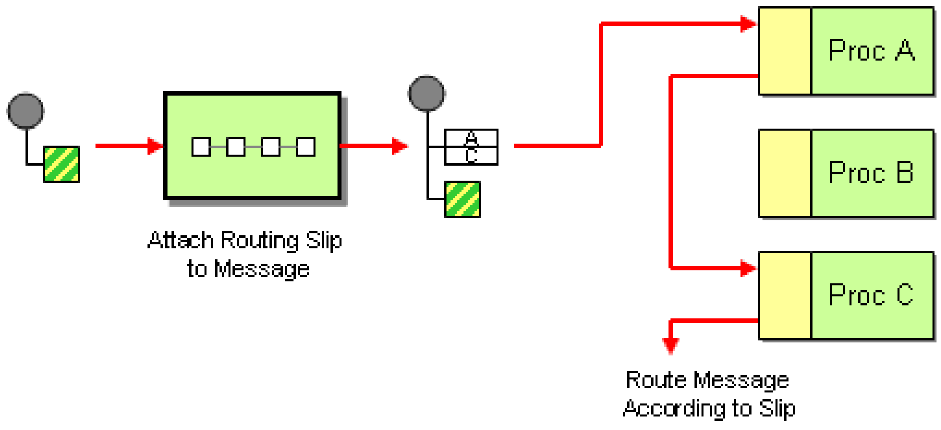

Attach Routing Slip to Message
A
C
Proc A
Proc B
Proc C
Route Message According to Slip

Figure 3.14. *Itinerary-based routing [63].*

Itinerary-based routing contributes to the distributed nature of the ESB. The itinerary details are stored as message metadata and carried with the message across the bus. There is no centralised rules engine to refer back to for each step in the process, thus the different parts of the ESB can operate independently. [24]

An itinerary represents a business process definition. They are best fitted for stateless processes that contain a limited number of steps (so-called microflows). For more complex or long-running transactions, a specific orchestration engine can be added to the ESB as a service. [24] The ESB helps to solve the SOA issue of having to code the find-bind-invoke sequence into clients [23]. The itineraries describe the set of services that should be invoked [87, p.651]. Thus, identifying and locating the next service in the chain, binding to it, and invoking it, are all steps performed by the ESB and the sequence can be altered through a change in configuration [23].

3.6 Software solutions for integration

The integration concepts and architectures explained earlier can be implemented in various ways. It is possible to develop an integration solution from scratch, entirely with in-house coding. However, the amount of effort required and the related cost are usually prohibitive for anything but most trivial solutions. Usually it does not make sense to build integration that way, starting from the very basics. At the other end of the spectrum, integration can be bought as a service.

A common choice is in between those two extremes: a software vendor offers an off-the-shelf integration product, and a customer-tailored solution is built on top of it. This allows the customer to benefit from the experience the vendor has gained through spending significant amounts of time and money to develop these products. Most of the platforms require somewhat high levels of expertise (often being hired from outside).

Many large software companies have EAI and ESB offerings, for example, IBM WebSphere, Oracle ESB, Microsoft BizTalk Server, and TIBCO ActiveMatrix Service Bus. Examples of open-source options are Mule ESB, JBoss ESB, and Open ESB, just to name a few. All of these have similarities, but naturally no two are exactly alike. The product used in this project, Microsoft BizTalk Server, is introduced in Chapter 5.

3.7 Middleware and SOA performance evaluation

Considering the number of available options for integration, it is not a trivial task to determine the best possible solution. Basic understanding of the fundamental concepts (such as EAI and ESB), what they have to offer, and how they differ from each other, is a good starting point. Specifying the core integration needs of the project or organisation is also important. Vendors can have different definitions for what each “buzzword” means. This may create confusion and it is difficult to make an informed decision. Understanding at least basics of the architectures and determining the requirements of the project makes it easier to accurately judge which platform might suite the needs. This seems like stating the obvious, but cannot be overemphasised.

Such decision-making is, however, based on approximate qualitative analysis at best. For more detailed understanding, certain measurable criteria should be defined, and then evaluated against. Things to consider are such as latency, throughput, security, availability, and so on. These are often measurable qualities, and acceptable values or ranges can be defined and tested against. Ideas for testing and selection methodology could be derived from what is proposed in, e.g., [22;141].

However, actual testing can be very complicated. The performance of the platform depends significantly on the used hardware configurations. Further, the distributed nature means that there are externalities, e.g. the general performance of the network affects the results. Performance problems might have multiple causes, and identifying and isolating each can prove difficult if not impossible. Building a full-scale environment for testing would be costly, more so as it would have to be built for each platform included in the evaluation. Thus, performance estimates based on modelling, rather than testing, are likely more attainable.

Results from tests performed by others can also offer guidance. An example case of testing middleware platform for automation use can be found in [123]. The source offers some insight on using middleware in a scenario that includes hard real-time requirements. However, Microsoft BizTalk, the platform used in this work, is not optimised for low-latency scenarios: it aims to maximise throughput. It may be possible to provide low-latency solutions with sufficient consistency, but that would sacrifice many of the things the platform is planned to do well. [37, p.24]

This work tries neither to offer exact evaluation criteria, nor to test a platform (let alone multiple platforms) against such criteria. That is well beyond the scope of this work. This discussion aims more for offering food for thought and things to consider.

SOA performance evaluation is challenging as well. The performance of SOA falls into two broad categories: the performance of an individual service, and that of the composite services together [33, see 86]. Testing an individual service is rather straightforward, and there are well established methods and tools available. Testing the performance of services integrated by an ESB is far more complex. For example, in a service composition scenario, one service might expect high volumes of traffic from a specific consumer, while another expects high reuse. Thus, it is complicated to analyse the workload characteristics of the composition. The ESB also introduces processing overhead, in addition to the atomic service overheads. To analyse the performance as a whole, understanding of the individual services as well as the ESB characteristics is required. [86]

3.8 Information security aspects

Having information and functionality locked in isolated silos is a major usability issue, but at least it is beneficial for information security. After all, it is hard for attackers to access something that even intended users cannot access. As the integration concepts develop, new information security issues arise, and new security methods are necessary.

A traditional EAI hub has its benefits, but it also introduces obvious security issues. It is a central point for integration functionality, monitoring, and logging of information. As a single point of failure, and (possibly) a centralised location for data, it is a target for attacks. A positive aspect is that it makes centralised security controls easier.

The Publish-Subscribe messaging pattern has security implications as well. In traditional systems, identifying (authenticating) various parties plays a key role in security. Authentication between two parties directly contradicts with the loosely coupled nature of pub-sub, where publishers and subscribers are unaware of each other. Delegating some aspects of trustworthy interaction to the hub requires that the infrastructure as a whole is trusted. [49] For each case, it should be carefully considered whether this assumption holds, both currently, and in the future. Securing a multi-domain pub-sub system using role-based access control (RBAC) is discussed in detail in [11].

The move towards an ESB based SOA is a major paradigm shift and probably the single biggest cause of changes in the information security aspects within the scope of this work. Ensuring a secure infrastructure in a service-oriented environment is difficult and there is no standard information security framework for SOA [25]. Successful SOA means lower barriers for reuse, and in the progress it makes application, technology, and enterprise boundaries insignificant. This clearly has major effects on security. [76]

Information security in the context of traditional applications is generally well understood. There are many techniques, best practices, etc., available to secure applications against common threats. Figure 3.15 illustrates a traditional approach to application security. [76]

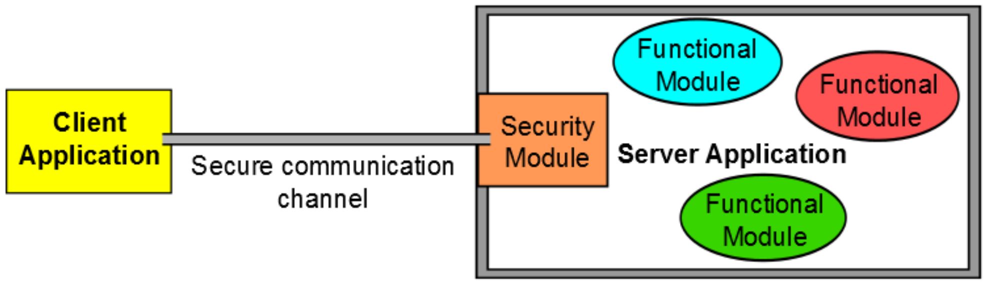


Figure 3.15. *Traditional application security approach, adapted from [76].*

A single server application has several functionalities that reside within clear borders. Access is provided through a secure channel, and security decisions are centrally handled by the application’s security module. This approach works well with traditional application architecture. The problem is that SOA makes many important security practices ineffective. Their use might even become counterproductive in SOA implementations. [76]

In comparison to Figure 3.15, three different server applications operating in a service-oriented environment are illustrated in Figure 3.16.

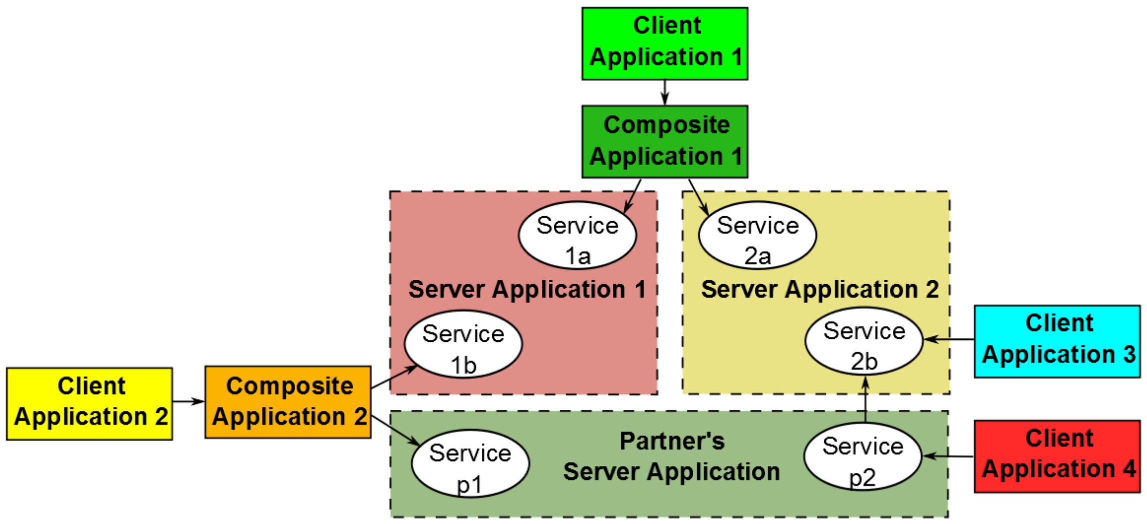


Figure 3.16. *Traditional application security approach is inadequate for SOA, adapted from [76].*

The security boundaries do not exist anymore. The client applications can be composed of any combination of services provided by the server applications. None of the applications control or have a complete view of the security model. A composite application may invoke services located in a completely different domain, like Composite Application 2 in Figure 3.16 does. [76] This serves as an example of how a common security method, consulting a centralised domain repository for authentication, does not work in SOA [25].

Service developers cannot know all the possible ways that the services might be invoked: Service 2b in Figure 3.16 is invoked by Client Application 3 and Service p2. Use of secure channels is also more complicated. For example, part of the information that Client Application 4 submits to Service p2 might be intended only for Service 2b. Thus, a secure channel between Service p2 and Client Application 4 is not sufficient for secure communications. These are but a few examples. The functional and non-functional aspects of information security and the insufficiency of traditional security approaches for SOA are discussed in detail in [76].

The problem is that the very principles that service design is based on directly contradict with information security [25;76]. A new way of looking at security is needed, and it has to be in alignment with the SOA design principles, otherwise the benefits of SOA will be lost. As was stated, SOA, in general, is about architecture rather than application development. Similarly, SOA security should focus on securing the architecture instead of a single application [76].

As a solution for the SOA information security challenges, three new security approaches are proposed in [76]: message-level security, security as a service, and policy-driven security. The discussion is continued in [25], where four components (namely, SOA information security governance, management, model, and policy information security framework) are proposed as a guide towards a SOA information security framework.

4 INTEGRATION LANDSCAPE: SMART GRID AND THE DISTRIBUTION DOMAIN

Smart Grid visions and plans promise many benefits and new usage scenarios. To successfully deliver these, efficient and secure integration of various information systems is required. This chapter will introduce the most important information systems that a DSO uses within the domain. Standards and guidelines play an important role when the goal is to achieve greater levels of interoperability. Various organisations have extended great effort in this field, and the key interoperability, integration, and information security standardisation work is introduced. Data flow diagrams are explained, as they are a good tool for analysis and design of both integration and information security.

The Smart Grid is a challenging integration environment for many reasons: it is complex, very heterogeneous, extremely large, and critically important. The landscape is illustrated in Figure 4.1, which shows the various information systems and connections. It is a conceptual reference, not a design diagram that defines a solution and its implementation. It is divided into domains similarly as Figure 2.1. [102]

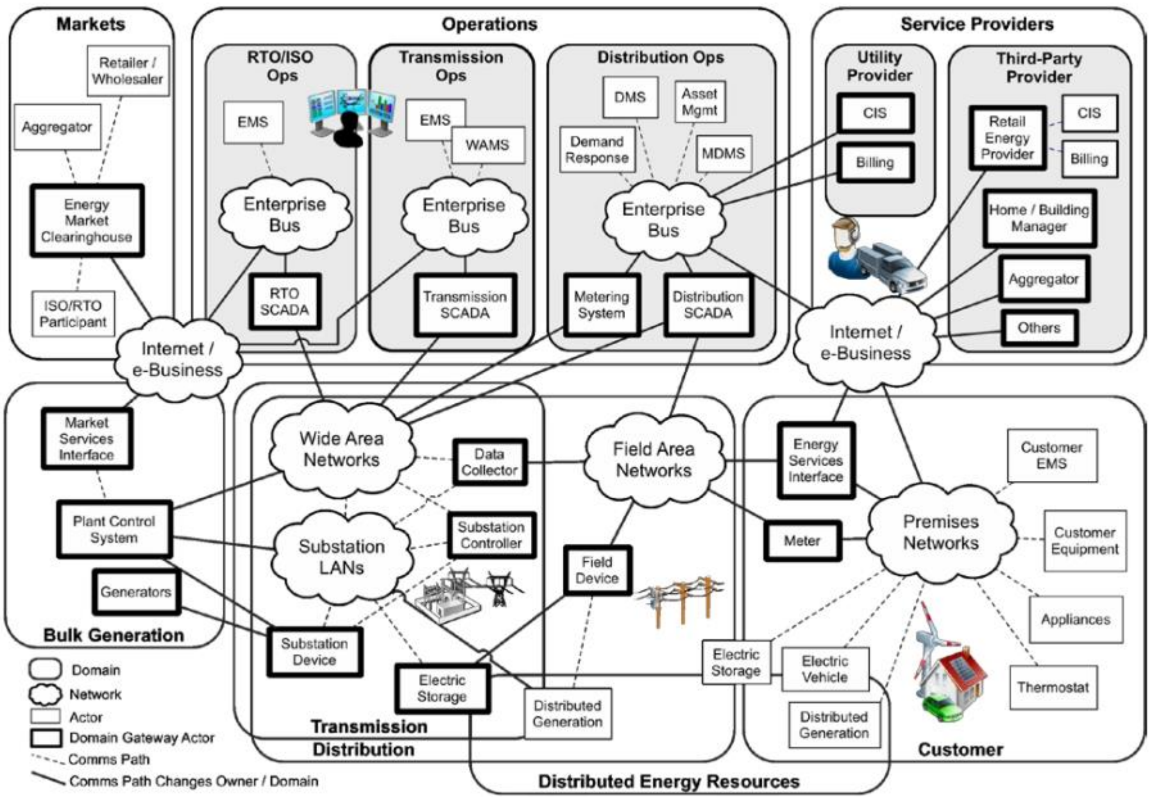


Figure 4.1. *Conceptual reference diagram for Smart Grid information networks [102].*

The electrical network can be divided into primary and secondary networks. Primary network consists of the components and wires that transmit the electricity. The secondary network consists of the devices and software that are used to operate and manage the primary network. Another name for the secondary network is distribution automation [83]. Figure 4.1 partly illustrates this division.

The distribution and transmission domains consist of the primary network components and relatively low-level secondary network components (e.g., substation controllers and field devices). Data from these components is usually aggregated before it enters the integration solution (the enterprise bus in the figure). More complex information systems, such as Distribution Management Systems (DMS), Supervisory Control and Data Acquisition (SCADA) systems, Network Information Systems (NIS), and Customer Information Systems (CIS) are located in the domains of operations and service providers. Figure 4.2 illustrates the key functions within the operations domain.

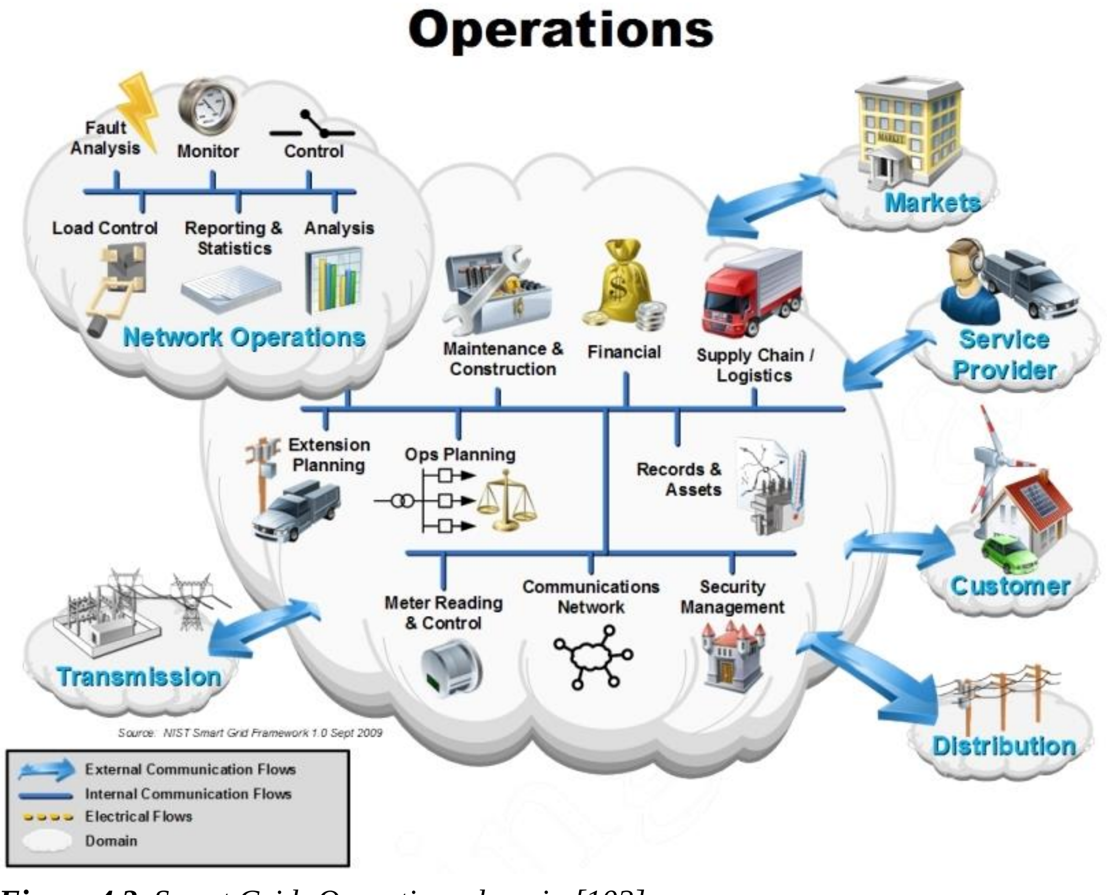


Figure 4.2. *Smart Grid: Operations domain [102].*

This work concentrates on the higher levels of control centre and company-level automation, and integration of the related information systems. Thus, more accurately, the focus is on the operations supporting the distribution domain, rather than the distribution domain itself. In this work, the term “distribution” generally covers both the distribution and operations domains, just as a DSO has control over both of these. The term is used here to separate this discussion from the concerns of generation, transmission, and customer domains. Finally, it should be pointed out that emphasising one area is no reason to ignore the fact that all the domains are interconnected.

4.1 DSO information systems

The DSOs have a wide variety of tasks to perform: network monitoring and operation, metering, customer billing, network planning, network maintenance, and so on. Figure 4.3 shows main operations performed by a DSO, along with the required information systems and information exchange. Many operations or processes have a dedicated software system (that likely interacts with other systems). These are provided by various vendors. A DSO likely uses a combination that consists of arbitrary number of systems from different providers. This leads to complex, heterogeneous environments. The benefits of integration are obvious in such environments.

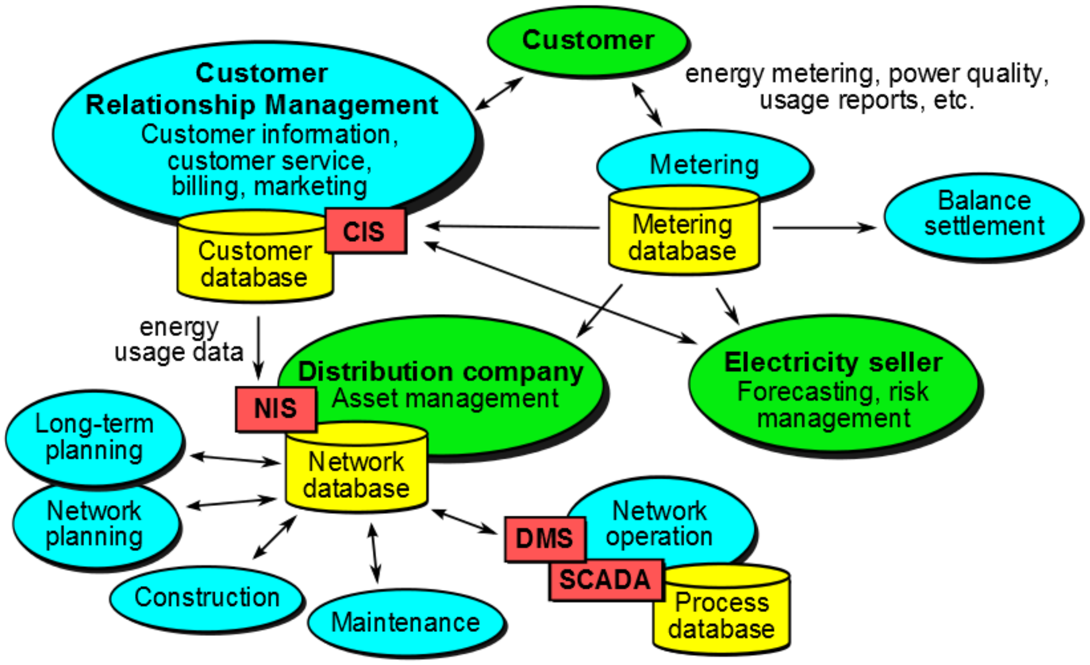


Figure 4.3. *Examples of information exchange needs between DSO business functions and software systems, adapted from [137;138].*

The challenges in integrating these systems are in no way new, and have been studied throughout the years at TUT, in Finland, and worldwide (e.g., [7;113;114;148]). More than a decade ago, the integration of applications was already stated as a long-term goal for utilities [7]. The industry clearly develops at a rather slow pace (much like the field of industrial control). There are some recent studies covering this issue from the point of view of Finnish DSOs [52;81;137].

Currently, most DSOs have implemented some level of integration between their IT systems. However, the solutions are often point-to-point based and product and company specific, each DSO having different implementations. [81;137] For example, according to [81], there are 89 DSOs in Finland, each having different IT system combination and level of integration [81, p.59]. There are no standard interfaces or standard information models in use [74]. Generally, the interfaces should be based on a standard, canonical data model. This is not a new idea either, and has been suggested, for example, in a 2004 article [120]. The Common Information Model (CIM, see Chapter 4.2.2) is intended to be used as such data model for the utilities industry.

Figure 4.4 gives an example of the required connections between various systems.

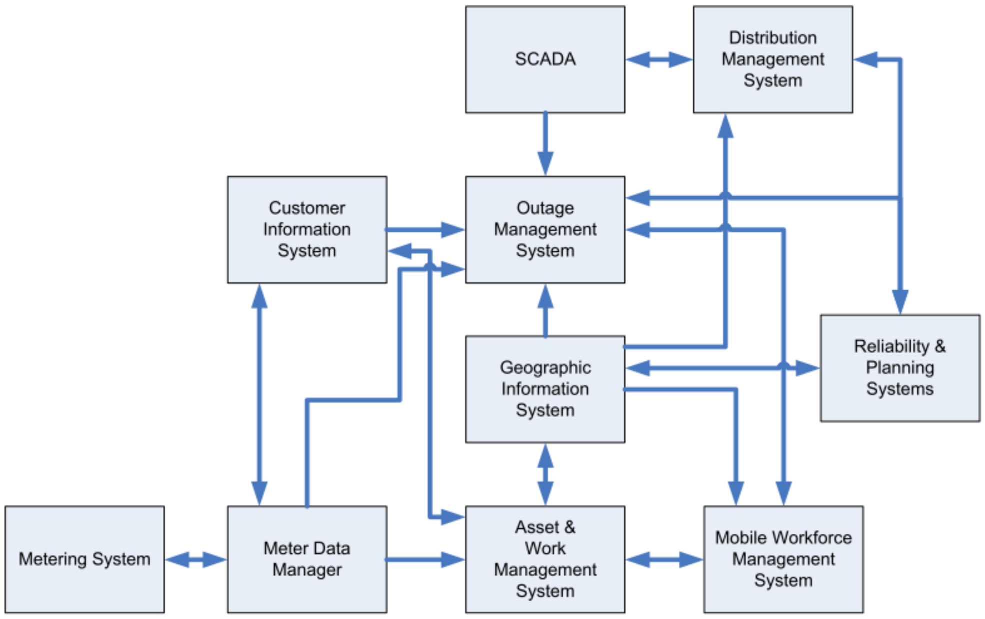

SCADA
Distribution Management System
Customer Information System
Outage Management System
Reliability & Planning Systems
Geographic Information System
Metering System
Meter Data Manager
Asset & Work Management System
Mobile Workforce Management System

Figure 4.4. *Example distribution systems interconnections [39].*

Connections shown are logical, and the figure clearly shows the downsides of creating the actual integration solution following this point-to-point logical topology. Adding new applications would be difficult. Both the EAI and ESB type integrations have been offered, as well as implemented, as a solution.

Figure 4.5 illustrates the conceptual idea of a control centre ESB, showing the environment and some of the various information systems that will be integrated using this ESB. The lower-level systems that do not directly connect to the ESB have an important role in the DSO architecture, but are not the main concern of this work.

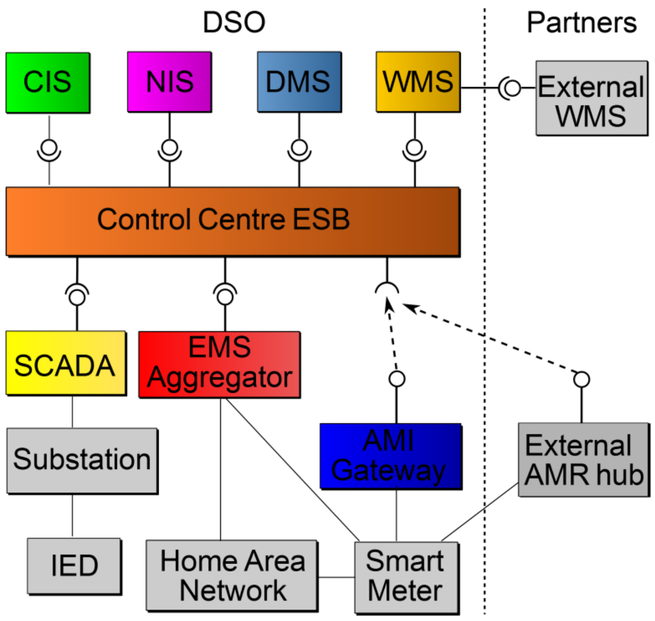


Figure 4.5. *The concept of control centre ESB and sample DSO IT systems.*

The core systems in Figure 4.5 are usually owned and operated by the DSO, but many business functions are outsourced to service providers. This is likely a growing trend. The service provider systems will need to be integrated to the control centre ESB as well, which clearly has security implications. Most important systems connecting to the ESB are introduced next. For further information, see, e.g., [42;81;104].

4.1.1 Supervisory Control and Data Acquisition (SCADA)

Supervisory Control and Data Acquisition (SCADA) is one type of an Industrial Control System (ICS). The term is not specific to electrical utilities; it is commonly used in many fields such as industrial processes, water treatment, or building automation.

In the electrical network SCADA is used for communicating with measurement and control devices within the network. SCADA system has the following attributes [38, p.143-144]:

1. Data acquisition: collecting data describing the operating state of the system and passing it to the control centre (in a near real-time manner).
2. Monitoring, event processing and alarms: comparing measured values to normal values and limits, and detecting changes in status, alarming operator of any critical events.
3. Control: manually initiated or automatic (event or time-triggered) control actions of, for example, specific devices (e.g. a circuit breaker or tap-changer).
4. Data storage, event log, analysis and reporting: data update overwrites the real-time measurements in the database, so time-tagged data is stored in the historical database at periodic intervals, for future use.

What differentiates SCADA from most of the other systems listed here is the fact that it deals with process information and often controls the operation of devices, with more or less in a real-time manner. Thus, it has stricter requirements for performance and security than most of the other systems.

4.1.2 Distribution Management System (DMS)

The Distribution Management System (DMS), along with SCADA, is a crucial information system within the DSO control centre. It is a collection of applications used to monitor, control and optimise the performance of the distribution system. It is an attempt to manage the complexity of the system. [38, p.141]

The goal of a DMS is to enable a smart, self-healing distribution system and to provide improvements in supply reliability and quality, and efficiency and effectiveness of system operation. Usually a DMS is able to combine the more static network data from, for example, the NIS, and dynamic measurement data provided by SCADA.

The DMS was referred to as a “collection of applications”, and its boundaries are not explicitly defined. Depending on the product vendor, the DMS may offer different things. The core functionality stays the same, but certain functions are sometimes offered as part of the DMS, sometimes as a part of some other system. For example Network Information System (NIS) functionalities could be combined to the DMS, thus eliminating the need for separate NIS software.

4.1.3 Network Information System (NIS)

The term Network Information System (NIS) is often used in Finland. Elsewhere, the same system is a combination often referred to as Automated Mapping/Facilities Management/Geographical Information System (AM/FM/GIS). [81]

The NIS is a network database that contains data of various network components. This includes electrotechnical data, other technical data, location information and background maps, and condition data. Information about network topology (the status of switching) is stored in the NIS. It can perform various calculations, for example for load flow, fault current, and reliability. It also supports network planning applications. [74]

4.1.4 Customer Information System (CIS)

As the name implies Customer Information System (CIS) stores and processes information about customers. It handles various things such as holds an inventory of meters and their locations, processes billing for customers, delivers bills, identifies losses, identifies customers affected by outages, informs customers of scheduled system maintenance, and helps complain handling and customer service personnel [38, p.145].

In a sense it is not a core part of the distribution automation, but it serves an important purpose as other systems access its information. In the Smart Grid scenarios, it is participating in many new use cases. It should be integrated to the ESB, and likely its importance will only increase in the future scenarios.

4.1.5 Advanced Metering Infrastructure (AMI)

In the heart of the Advanced Metering Infrastructure (AMI) is the smart meter. It is the component of the Smart Grid that is most familiar to the customer, and the AMI is more accurately part of the customer domain, but it is tightly connected to the distribution operations as well. The rollout of smart meters in European Union is mandatory, and the meters have induced active public discussion.

The terminology is somewhat confusing. Smart metering is used as an umbrella term that constitutes of Automated Meter Reading (AMR), Automated Meter Management (AMM), and Advanced Metering Infrastructure (AMI). Smart metering is seen as an essential part of the Smart Grid [74].

The AMR system offers many new possibilities. For example, it is envisioned as an extension of SCADA and DMS for controlling and monitoring also the fuse protected networks, especially LV-networks [74]. However, the smart meter is only a portion of the entire metering infrastructure. As there can be hundreds of thousands of meters installed within a DSO’s network, it is obvious that the metering data needs to be somehow aggregated before it can be transferred into higher-level information systems.

4.2 Smart Grid and integration standardisation efforts

Standards are an essential part of engineering work. Large projects, such as the Smart Grid, would be impossible without standardisation. This is a brief look at some of the most important organisations that provide standards and guidance, and to the work they have done to enable the Smart Grid. The term “standard” is used here very liberally, as these are more of references, guidelines, frameworks, recommendations, and so on. However, they are crucial to Smart Grid development, and many either are, or will likely become, standards or de facto standards. For simplicity, they are all referred to as “standards”.

Many of the envisioned Smart Grid scenarios require that existing systems can work efficiently together, and access information and functionality within other systems. Thus, many standards aim for greater interoperability, defined as the “ability of two or more networks, systems, devices, applications or components to exchange information between them and to use the information so exchanged” [47, p.5]. The goal of greater interoperability is in no way unique to the electric system.

4.2.1 Roadmaps, frameworks, guidelines, and recommendations

Various organisations offer roadmaps that list the most important Smart Grid standards. This again illustrates the complexity of the topic. In addition to the ones introduced here, many other organisations worldwide offer similar documents.

Notable examples are the following: The International Electrotechnical Commission (IEC) has a Smart Grid Standardization Roadmap [70]. The Institute of Electrical and Electronics Engineers (IEEE) offers guidance for interoperability as well as information security in [64]. The European Electricity Grid Initiative (EEGI) provides a roadmap and detailed implementation plan in [46].

NIST: list of interoperability standards for Smart Grid

The U.S. National Institute of Standards and Technology (NIST) have done significant research in the field of Smart Grid standards. A good starting point to understand the Smart Grid standardisation in general is the “NIST Framework and Roadmap for Smart Grid Interoperability Standards”.

The first release (2010 [101]) provides a list of 75 standards that NIST sees as most important for the development of Smart Grid. Most of the subsequent standards listed in this chapter are on the list. It introduces the conceptual reference model (shown in Figure 2.1 and Figure 4.1). Priority action plans to fill gaps in standardisation are also suggested, and cyber security aspects are covered. The second release (2012 [102]) of the same document contains updates of the achieved improvements, and cyber security aspects have a dedicated chapter.

GWAC: interoperability context-setting framework

The GridWise Architecture Council (GWAC) also originates in the United States: it was formed by the Dept. of Energy to promote and enable interoperability in the Smart Grid. Publications are available from [54]. The "GridWise Interoperability Context-Setting Framework" illustrates interoperability on a conceptual level. The concept of "distance to integrate" is illustrated in Figure 4.6. The greater the customisation efforts and manual work required for two systems to be interoperable, the greater the distance to integrate [53].

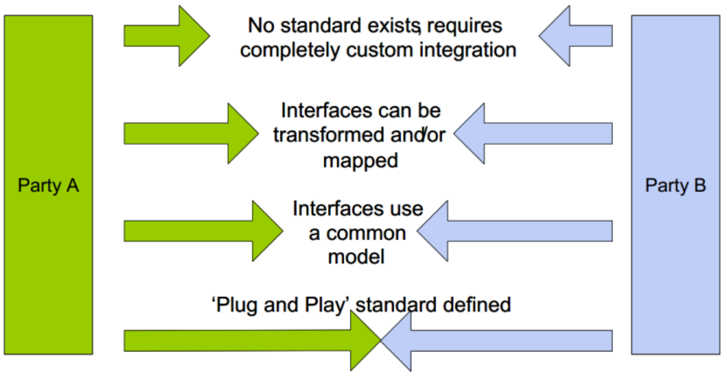

Party A
No standard exists requires completely custom integration
Interfaces can be transformed and/or mapped
Interfaces use a common model
'Plug and Play' standard defined
Party B

Figure 4.6. *Distance to integrate [53].*

The distant is non-existent with "Plug and Play" interoperability, a concept familiar from e.g., consumer electronics. For complex systems, Plug and Play might not be a practical goal, but standards and best practices can help to minimise the distance. The end goal is, of course, to lower the installation and integration costs and allow individual components and systems to be interchangeable (with reasonable effort). [53]

The GWAC framework (referred to as the "GWAC stack") introduces eight categories of interoperability, which are further grouped into organisational, informational, and technical aspects (Figure 4.7).

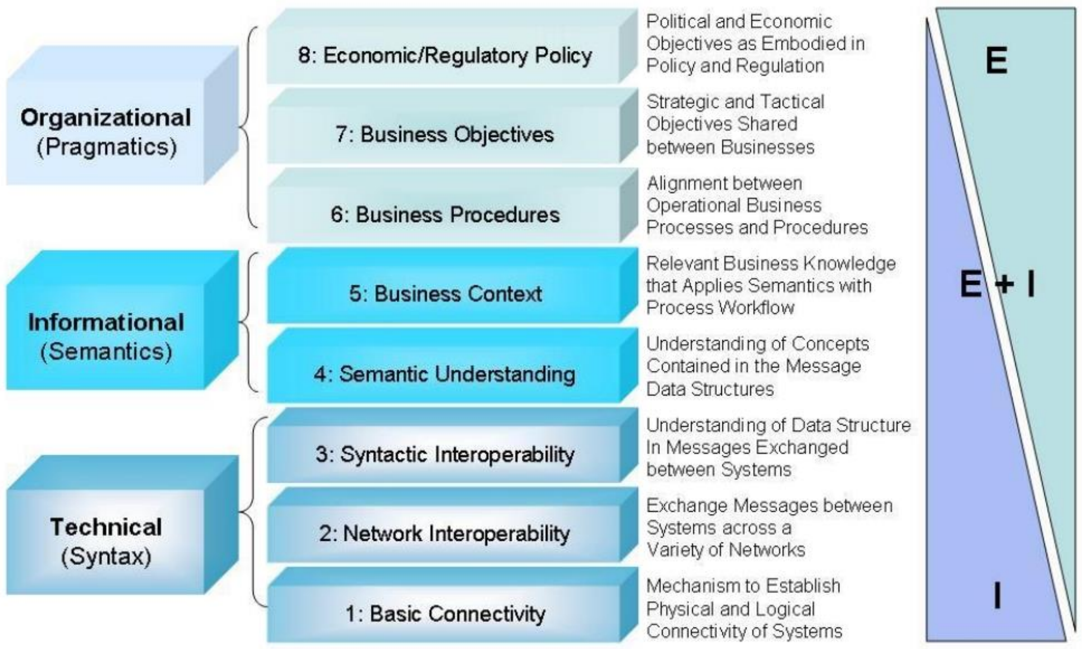

Organizational (Pragmatics)
Informational (Semantics)
Technical (Syntax)
8: Economic/Regulatory Policy
7: Business Objectives
6: Business Procedures
5: Business Context
4: Semantic Understanding
3: Syntactic Interoperability
2: Network Interoperability
1: Basic Connectivity
Political and Economic Objectives as Embodied in Policy and Regulation
Strategic and Tactical Objectives Shared between Businesses
Alignment between Operational Business Processes and Procedures
Relevant Business Knowledge that Applies Semantics with Process Workflow
Understanding of Concepts Contained in the Message Data Structures
Understanding of Data Structure In Messages Exchanged between Systems
Exchange Messages between Systems across a Variety of Networks
Mechanism to Establish Physical and Logical Connectivity of Systems
E
E + I
I

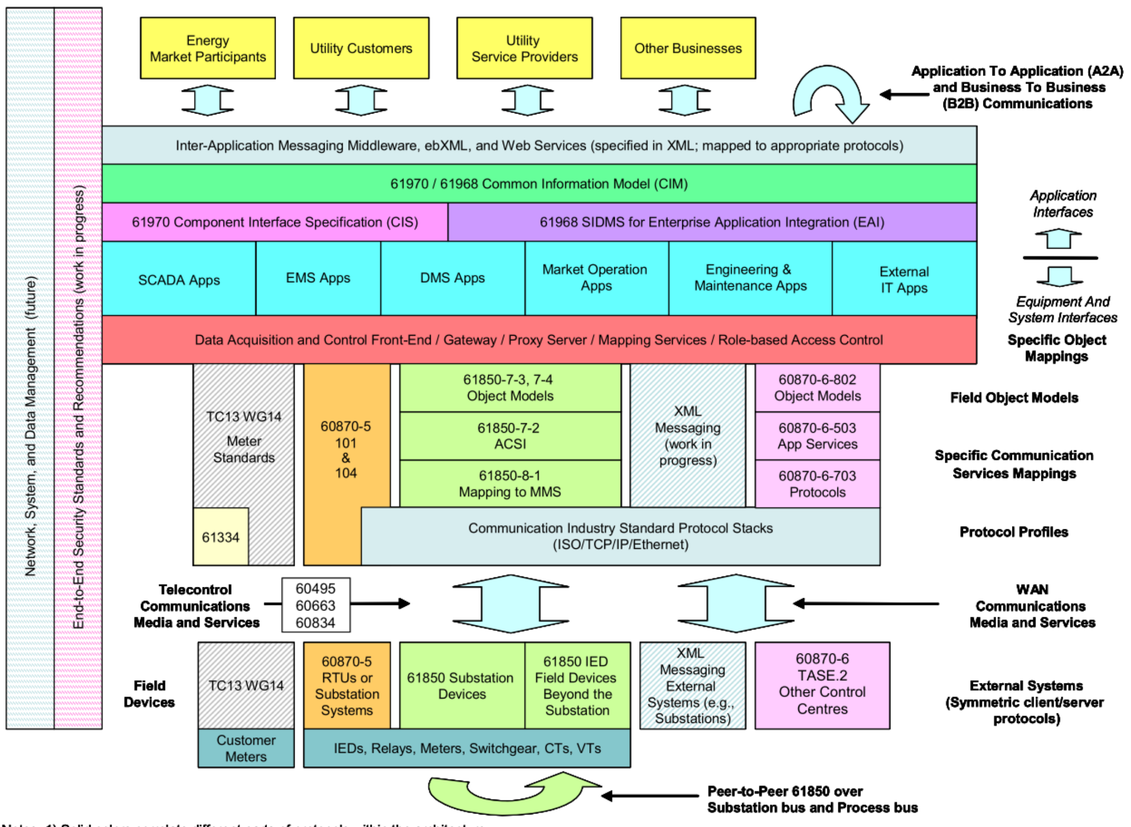
Figure 4.7. *GWAC: Interoperability Framework Categories [53].*

The idea of context-setting framework is to organise concepts and terminology so that problems can be identified and discussed effectively (i.e. “to put everyone on the same page”). The E+I infrastructure depicts the interconnected nature of electricity and information technology. The organisational (top) layers (pragmatics) are concerned with the management of electricity. At the bottom, the technical layers deal with communications, networking, and syntax issues of IT. The semantics, the informational layers in the middle, are used for transforming information into knowledge that supports the electricity related businesses. [53]

The issues that this work is concerned with are mainly located on layers three and four of the stack. The framework also identifies another dimension: the cross-cutting issues that affect all the layers. For example, security, privacy, and quality of service are concerns that cross-cut all the layers, and are important in terms of this work.

IEC: TC57 reference architecture

The IEC Technical Committee 57 provides reference architecture for Smart Grid, as shown in Figure 4.8 (and in Figure B.1, Appendix B). The charter for TC 57 is:”Power System Management and Associated Information Exchange”.



Network, System, and Data Management (future)
End-to-End Security Standards and Recommendations (work in progress)
Energy Market Participants
Utility Customers
Utility Service Providers
Other Businesses
Application To Application (A2A) and Business To Business (B2B) Communications
Inter-Application Messaging Middleware, ebXML, and Web Services (specified in XML; mapped to appropriate protocols)
61970 / 61968 Common Information Model (CIM)
61970 Component Interface Specification (CIS)
61968 SIDMS for Enterprise Application Integration (EAI)
Application Interfaces
SCADA Apps
EMS Apps
DMS Apps
Market Operation Apps
Engineering & Maintenance Apps
External IT Apps
Equipment And System Interfaces
Data Acquisition and Control Front-End / Gateway / Proxy Server / Mapping Services / Role-based Access Control
Specific Object Mappings
TC13 WG14 Meter Standards
61334
60870-5 101 & 104
61850-7-3, 7-4 Object Models
61850-7-2 ACSI
61850-8-1 Mapping to MMS
XML Messaging (work in progress)
60870-6-802 Object Models
60870-6-503 App Services
60870-6-703 Protocols
Field Object Models
Specific Communication Services Mappings
Communication Industry Standard Protocol Stacks (ISO/TCP/IP/Ethernet)
Protocol Profiles
Telecontrol Communications Media and Services
60495
60663
60834
WAN Communications Media and Services
Field Devices
TC13 WG14
60870-5 RTUs or Substation Systems
61850 Substation Devices
61850 IED Field Devices Beyond the Substation
XML Messaging External Systems (e.g., Substations)
60870-6 TASE.2 Other Control Centres
External Systems (Symmetric client/server protocols)
Customer Meters
IEDs, Relays, Meters, Switchgear, CTs, VTs
Peer-to-Peer 61850 over Substation bus and Process bus

Notes: 1) Solid colors correlate different parts of protocols within the architecture.
2) Non-solid patterns represent areas that are future work, or work in progress, or related work provided by another IEC TC.

Figure 4.8. *The IEC Reference architecture [70;71].*

The reference architecture shows how the various IEC and other standards relate to each other. This again indicates the complexity of the whole Smart Grid. For our purposes, most interesting are the 61970 and 61968 standards that constitute the Common Information Model (pictured as the bright green box in Figure 4.8).

4.2.2 IEC: Common Information Model (CIM)

Generally, the idea of an information model is to describe a problem domain without constraining how that description is implemented [39]. The Common Information Model, CIM, was originally developed by the Electric Power Research Institute (EPRI), and has been adopted by the IEC. The goal of CIM is to provide a canonical data model, a common vocabulary, for utilities industry. It aims to increase interoperability so that various systems can exchange information regarding the status and configuration of a network. The IEC is responsible for maintaining and developing the CIM model (currently, the formal definition uses the Unified Modelling Language, UML). [40;91]

The CIM User Group describes CIM as “an abstract information model that provides data understanding through the identification of the relationships and associations of the data within a utility enterprise” [140]. When the data is better understood (through the use of a common vocabulary), it becomes easier to exchange data models and messages and integrate applications (intra- and inter-enterprise) [140].

The CIM is a set of standards, and the entire model is large and complex, as it needs to cover such a wide range of topics. The EPRI CIM primer states that the standards IEC 61970-301, 61968-11, and 62325-301 are collectively known as the CIM, and proceeds to list their three primary uses. These are 1) to facilitate the exchange of power system network data between organisations, 2) to allow the exchange of data between applications within an organisation, and 3) to exchange market data between organisations. [40, p.7].

The standards 61970 and 61968 are of most interest in this work, as the use cases are mainly based on these. These are defined as follows, in [71].

61968: “Standards for Distribution Management System (DMS) interfaces for information exchange with other IT systems. These include the distribution management parts of the CIM and eXtensible Markup Language (XML) message standards for information exchange between a variety of business systems, such as meter data management, asset management, work order management, Geographical Information Systems (GIS), etc.” [71]

61970: “Standards to facilitate integration of applications within a control center, exchange of network power system models with other control centers, and interactions with external operations in distribution as well as other external sources/sinks of information needed for real-time operations. These standards include the generation and transmission parts of the Common Information Model (CIM), profiles for power system model exchange and other information exchanges, and XML file format standards for information exchange.” [71]

Some of the important features of the CIM (listed in [71]) are as follows.

- The CIM is hierarchical.
- The CIM is normalised.
- The CIM is static.
- The CIM is modelled in UML.

- The CIM IEC standards documents are auto-generated using the electronic UML model.
- The CIM has a representation in XML.
- The CIM is in use in many production systems.
- The CIM is meant to contain classes and attributes that will be exchanged over public interfaces between major applications. [71]

The IEC 61968 standard provides an Interface Reference Model (IRM), illustrated in Figure 4.9, which divides the typical DSO operations into 14 business functions.

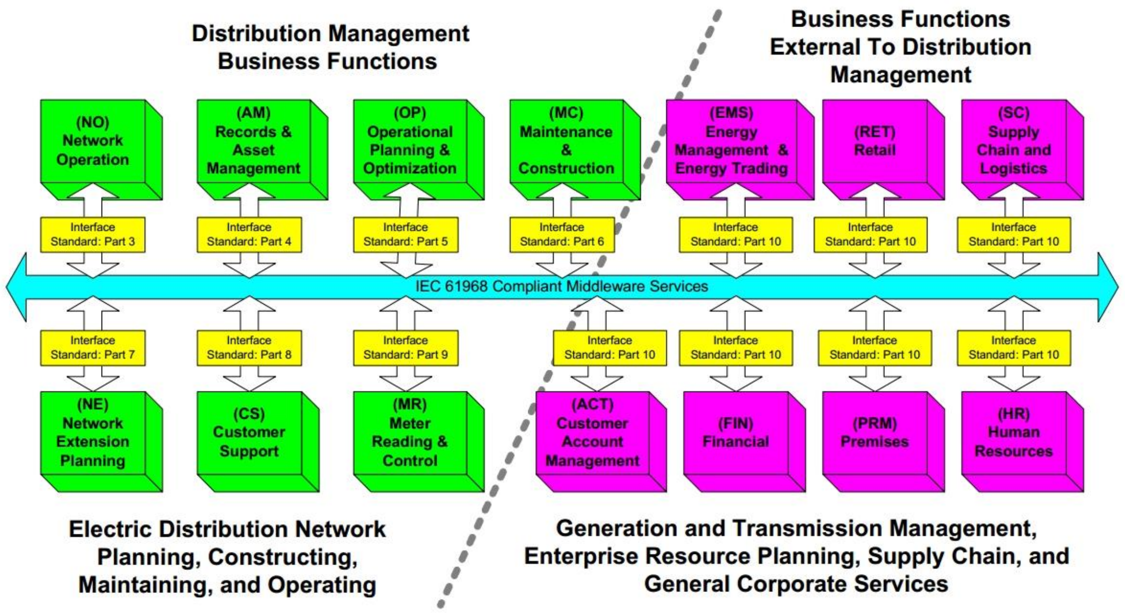

Distribution Management Business Functions
Business Functions External To Distribution Management
(NO) Network Operation
(AM) Records & Asset Management
(OP) Operational Planning & Optimization
(MC) Maintenance & Construction
(EMS) Energy Management & Energy Trading
(RET) Retail
(SC) Supply Chain and Logistics
Interface Standard: Part 3
Interface Standard: Part 4
Interface Standard: Part 5
Interface Standard: Part 6
Interface Standard: Part 10
Interface Standard: Part 10
Interface Standard: Part 10
IEC 61968 Compliant Middleware Services
Interface Standard: Part 7
Interface Standard: Part 8
Interface Standard: Part 9
Interface Standard: Part 10
Interface Standard: Part 10
Interface Standard: Part 10
Interface Standard: Part 10
(NE) Network Extension Planning
(CS) Customer Support
(MR) Meter Reading & Control
(ACT) Customer Account Management
(FIN) Financial
(PRM) Premises
(HR) Human Resources
Electric Distribution Network Planning, Constructing, Maintaining, and Operating
Generation and Transmission Management, Enterprise Resource Planning, Supply Chain, and General Corporate Services

Figure 4.9. *The IEC Interface Reference Model (IRM) [69, see 108].*

The IRM provides the framework for a series of message payload standards. The business functions are illustrated by the green and purple boxes, and these are connected by the CIM-compliant (ESB) middleware, pictured as a light blue bus. The yellow boxes represent standard interfaces, and each of these is described by a specific part of the IEC 61968 standard. These 61968 series standards define the use of XML for the exchange of information between the various systems defined in the IRM.

An effort similar to CIM is the MultiSpeak, coordinated by the National Rural Electric Cooperative Association (NRECA). The MultiSpeak is arguably more mature model than the CIM, but does not have as wide coverage as the CIM. It is North America centred, whereas the CIM is international. MultiSpeak and CIM have slightly different approaches, and there are efforts to harmonise these models. [103]

MultiSpeak will not be discussed here; more information can be found in [103]. Details of the CIM and how to apply it can be found in, e.g., [39-41;90;91;142].

4.2.3 W3C recommendations and integration

Many recommendations from the World Wide Web Consortium (W3C) [150] have become de facto standards and basic building blocks for integration (e.g., XML, Resource Description Framework RDF, Web Services, and their related recommendations). These recommendations are general and applicable for many fields, i.e. they are not Smart Grid specific. They form the foundation of many concepts and technologies used in this work. For example, many integration and SOA implementations, the CIM, and the BizTalk platform, all make use of the W3C recommendations.

The technologies based on these recommendations have become so ubiquitous that a general introduction is included in various books and other sources, and thus will not be repeated here. For more details, the official W3C recommendations, found in [149], are a good starting point, as well as books such as [111].

4.2.4 Smart Grid information security standards

Various organisations provide standards and guidelines dedicated to the information security aspects of Smart Grid and industrial control systems in general. These are partly overlapping, but often differ slightly in their approach, and a few are introduced here. The NIST offers both 800-series Special Publications, and the NIST Interagency Reports (NISTIRs), that are dedicated to control systems and the Smart Grid. Notable work has been done by the International Organization for Standardisation (ISO), the IEEE, the IEC, the European Network and Information Security Agency (ENISA) [48], the International Society of Automation (ISA), the (U.S.) Department of Homeland Security (DHS), and the North American Electric Reliability Corporation Critical Infrastructure Protection Committee (NERC CIPC), among others.

The NIST Special Publication SP 800-82 Guide to Industrial Control System (ICS) Security aims to provide guidance for securing ICSs. It explains shortly the specific nature of ICSs (compared to IT), and identifies some of the common threats and vulnerabilities. It also lists recommended security methods to mitigate the risks. [131] It is not intended to be a checklist for security, and does not cover the topic in detail, but it is a good starting point to familiarise oneself with the ICS information security aspects.

The NIST IR 7628 Guidelines for Smart Grid Cyber Security is the work of the Smart Grid Interoperability Panel (SGIP) Cyber Security Working Group (CSWG) [127]. It is offered as a three-volume set, with a separate introductory document, and intended to be a companion document to the NIST Framework and Roadmap for Smart Grid Interoperability Standards. The first volume describes the approach used to identify high-level security requirements, and presents a high-level architecture, and proceeds to more details [124]. The second volume is dedicated to the privacy issues and concerns of the customer, also providing recommendations [125]. The third volume lists the supporting analyses and references that were used in the development of the first two volumes [126].

In addition to these information security specific standards, most of the more general Smart Grid standards have a part dedicated for information security. The awareness about the issues is increasing rapidly. Yet, in the end, it should be noted that information security contributes only to a small part of the overall safety and security requirements.

4.3 Data flows within the distribution domain

Data flow diagrams (DFDs) are a good tool to help understand the information exchange between systems. DFDs are not included as part of the UML, but have been used in software and systems development for many decades. Other (UML) diagrams are useful tools as well, but DFDs are especially useful for integration purposes, when defining data flows between various systems.

DFDs are used to describe logical connections and data flows, independent of the used technology. The basic components, and guidance for drawing DFDs, are introduced here (for details, see, e.g., [36;55;115;132]). Further, example context diagram and data flows within a DSO are illustrated.

4.3.1 Data flow diagrams (DFDs)

A DFD shows information flows that go into, and out of, a system. It shows the sources and destinations of the flows, and stores of data within the system. A DFD is not concerned with the timing or sequence of the flows, the reasons why a flow occurs, or the technical implementation of the flow.

DFDs are not very strictly or formally defined, but certain rules help make them more understandable. A few widely used notations exist, yet the main components, shown in Figure 4.10, are the same in each notation: processes (functions), data repositories, external entities (inputs/outputs, sources/sinks) and data flows.

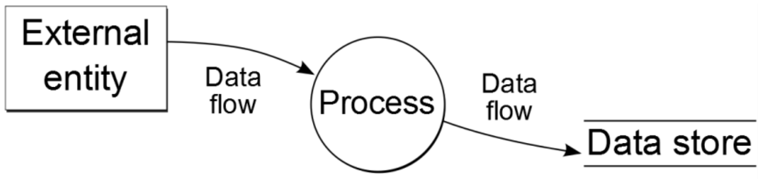


Figure 4.10. *Main components of a DFD.*

Process is where data is used or generated. Process labels should be verb phrases describing what the process does. External entities represent an external source, user or depository of the data. Labels should be noun phrases. Data is stored into and retrieved from data stores, which are internal repositories of data. Data flow (a connecting arrow) represents how data flows through the system. [115]

Importantly, DFDs are hierarchical; that is, a process can be decomposed to sub-processes. Figure 4.11 illustrates the idea of decomposition.

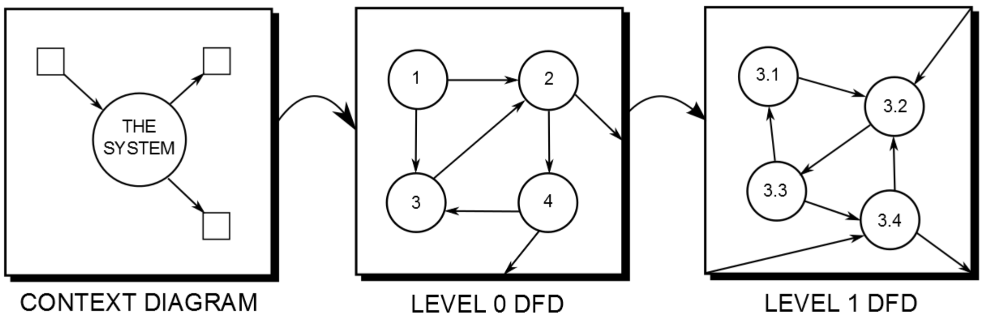


Figure 4.11. *Decomposition of DFDs, adapted from [132].*

The context-level DFD shows the entire system as a single process, which interacts with external entities. Internal organisation of the system is not shown. The diagram implicitly defines the system boundaries, as the external entities communicate with the system, but are not part of it, and the system has no control over them. The context diagram is decomposed to form the level 0 DFD, which shows how the system is divided into sub-systems (processes). Each of the subsystems deals with one or more of the data flows to or from an external system. Combined, the subsystems provide all of the functionality of the system as a whole. Level 1 DFD is again the result of decomposition of a process in the level 0 diagram. At the lowest level, processes can no longer be decomposed.

4.3.2 Example DFDs

Figure 4.12 illustrates a sample context diagram of an integration platform (EAI or ESB) within a DSO environment. As is often the case with context diagrams, it is so simple that it seems hardly worth the effort to draw it. However, it indicates where the borders of the system are. It shows how the systems that will be integrated are outside the scope of the ESB/EAI implementation itself.

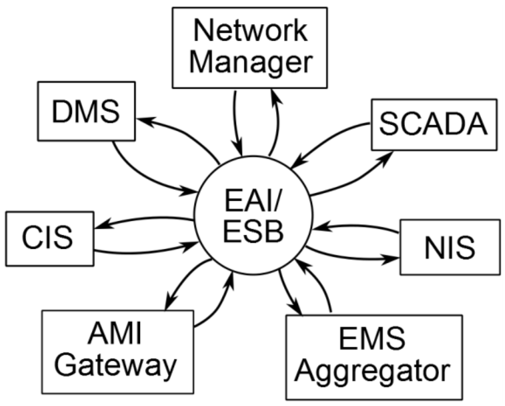


Figure 4.12. *A sample context diagram of an integration platform.*

In the case of the integration solution, the level 0 DFD is rather simple. The main purpose of the middleware is to transmit information between various endpoints. The process does involve many sub-processes, such as endpoint resolution and data format transformation. However, only one process, “route information”, is used in the level 0 DFD. It is also possible to break the routing process into its constituent parts. This could be used, for example, to illustrate the internal operation of the middleware. However, analysis at that level of detail is unnecessary for the purposes of this work.

Figure 4.13 illustrates sample flows that could occur within a DSO environment. This already shows enough details to help analyse the information content of each flow.

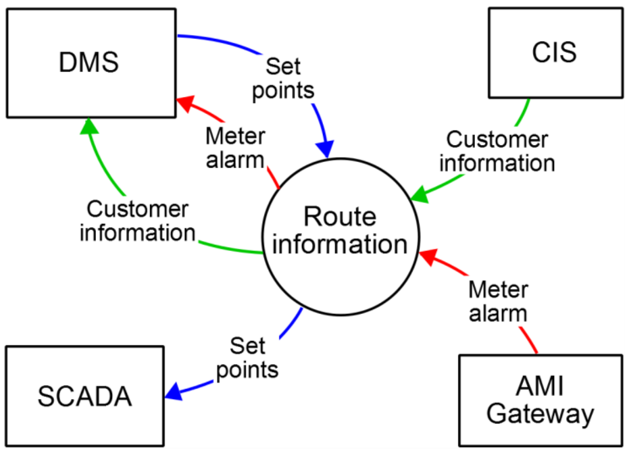


Figure 4.13. *Sample Level 0 DFD within a DSO environment.*

However, the traditional DFD functional decomposition is somewhat ineffective for the purpose of this work. Level 0 DFD offers sufficient level of detail for analysing the integration in general, thus further decomposition is not necessary. Yet there are countless types of data flows within the environment, and it would obviously not make sense to try to illustrate each of them in a single DFD. However, it makes sense to use the DFDs on a per-use-case analysis: the process “route information” is same for each use case, but the participating systems and flows vary.

4.4 Using DFDs in information security analysis

The integration system context diagram (Figure 4.12) is useful also as a starting point for the information security analysis. Information is mostly stored within the endpoint systems, and proper security measures for these systems are crucial. Yet the detailed operation of the endpoints is generally beyond the scope of this work. Excluding the external systems from the analysis leaves two types of components in the context diagram: the central node (whether an ESB or an EAI hub), and the arrows (data flows).

Securing the central node, the integration platform itself, is extremely important. There are general best practices, such as hardening the platform by removing or inactivating all unnecessary functionalities, and always using the most up-to-date software. The specific security measures and implementation vary depending on which integration software product is used. Most vendors offer product-specific integration platform security guides, which are likely the best source for detailed information.

Finally, the security requirements for the data flows need to be analysed. The data flows represent the essential functionality of the integration, and their security aspects are considered in this work. Depending on the scenario (use case), any or all of the C-I-A objectives could be important, and at varying degrees. Figure 4.14 illustrates a normal data flow and the most common types of threats (interruption, interception, modification, and fabrication). Each scenario shows how a different objective (C-I-A) could be compromised, and what countermeasures are generally available. These are a typical starting point for information security related discussions in the literature, for details see, e.g., [16;112;130].

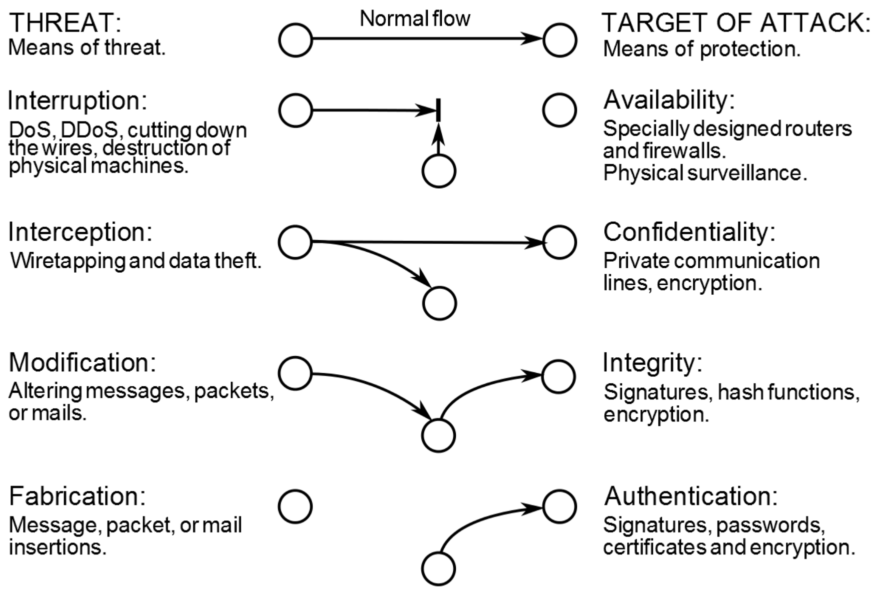

THREAT:
Means of threat.
Normal flow
TARGET OF ATTACK:
Means of protection.
Interruption:
DoS, DDoS, cutting down the wires, destruction of physical machines.
Availability:
Specially designed routers and firewalls.
Physical surveillance.
Interception:
Wiretapping and data theft.
Confidentiality:
Private communication lines, encryption.
Modification:
Altering messages, packets, or mails.
Integrity:
Signatures, hash functions, encryption.
Fabrication:
Message, packet, or mail insertions.
Authentication:
Signatures, passwords, certificates and encryption.

Figure 4.14. *Securing the flow of information, adapted from [129, see 105].*

Most middleware platforms provide the common countermeasures for each threat. For example, encryption aims to provide confidentiality, signatures are used for authentication, and so on. The implementation specifics vary for each platform, but conceptually, the methods are available. However, one issue that often arises in integration is the need to communicate with legacy systems. Even though the middleware platform offers encryption, it may have to communicate with an endpoint that does not support it.

Authenticity and accountability are important aspects of information security that many feel are not sufficiently covered in the C-I-A analysis [130, p.11]. Authenticity means that information can be verified and trusted. Accountability supports nonrepudiation: often, it is important to be able to prove that a certain event took place, and provide details of the event (such as participants in a transaction). Additionally, some data flows may have timeliness requirements requiring more detailed analysis. Thus, analysing the C-I-A requirements of data flows is a good starting point, but does not cover everything.

Analysing the security requirements is a task that requires co-operation of experts from various fields. Knowledge of the subject domain (electricity distribution) is necessary to understand the purpose of a use case or a data flow. Information security professionals have the expertise and mind-set to ask the right questions. Domain experts (electrical engineers) have the necessary understanding to answer those questions. Integration experts can design and implement the solution, once sufficient requirements for both functionality and security have been defined.

These steps are essentially about understanding the system, performing a risk analysis, and designing the implementation in a way that the risks can be mitigated. Risk analysis is a well-formulated and widely used method to improve information security. There are many formal approaches available which can be used as a framework and starting point for each specific risk analysis situation [112, p.526].

5 MICROSOFT BIZTALK SERVER

Microsoft's enterprise integration platform, BizTalk Server, is the integration product used in this project. Outside the field of enterprise integration, it is relatively unknown. Yet it claims the title for the most broadly deployed integration middleware technology of today, with more than 12 000 customers worldwide [87, p.xxii]. It has been described as "quite possibly the most advanced product produced by Microsoft to date" [121, p.9]. BizTalk has its roots in Enterprise Application Integration (EAI) and Business-to-business (B2B) integration, but over time it has developed into a highly complex and flexible software solution [97].

BizTalk Server is a mature software product. First release was BizTalk 2000, and as this work begun, BizTalk was in its seventh version (BizTalk 2010). During this work, BizTalk 2013 was released, first as a beta version, and later as an official release. Two versions are tested in this project: the latest stable release that was available at the beginning of the project (BizTalk 2010) and a beta version of the new BizTalk 2013. For the sake of simplicity, term "BizTalk" will be used, and a specific version is mentioned only when necessary.

5.1 Introduction to BizTalk

Figure 5.1 illustrates where BizTalk fits in an enterprise. It allows the (often massive) enterprise-scale systems to communicate with each other, within, as well as across, the company borders. Business rules guide the processes, and in addition to more technical reports, BizTalk offers comprehensive business process monitoring capabilities.

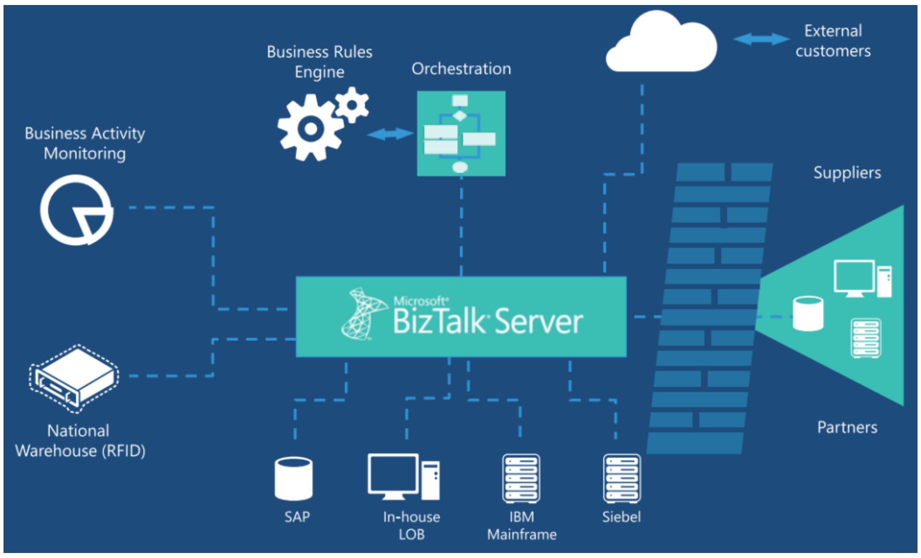


Figure 5.1. *Integration with BizTalk [18].*

It is not trivial to introduce something as complex as BizTalk and do it briefly, with simplicity and clarity, yet without overlooking any crucial properties. Throughout this introduction it is important to keep in mind that BizTalk is much more complex and has more components than is possible to cover in this limited space. BizTalk's versatility only adds to this complexity: it is difficult to pinpoint any typical BizTalk solution, as BizTalk can be used to build a multitude of different solutions, and in most any industry.

5.1.1 BizTalk components on a high level

BizTalk is, in essence, a message broker. It receives information from a source, often processes that information somehow, and sends it to one or more destinations. At its core, BizTalk functions as a publish/subscribe engine [56, p.7]. BizTalk's main components are shown in Figure 5.2. The messaging component is the heart of BizTalk, providing the ability to communicate with other systems. Continuing with the metaphor, if messaging is the heart of BizTalk, the orchestration engine is the brain. It is used to create and run graphically defined processes called orchestrations. [96]

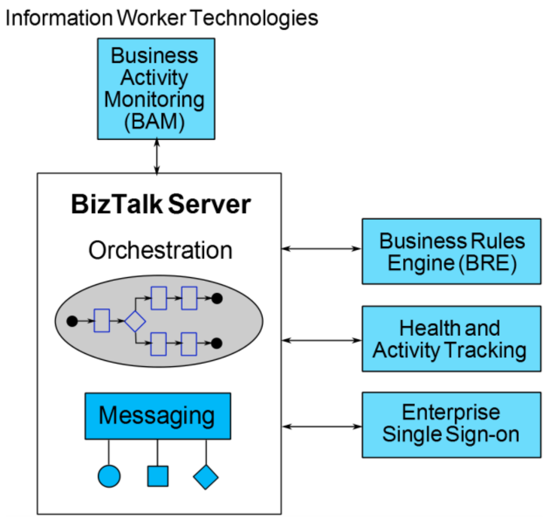

Information Worker Technologies
Business Activity Monitoring (BAM)
BizTalk Server
Orchestration
Messaging
Business Rules Engine (BRE)
Health and Activity Tracking
Enterprise Single Sign-on

Figure 5.2. *BizTalk Server core components, adapted from [96].*

The Messaging Engine receives inbound messages, parses them to identify their formats, and evaluates the message contents. The messages are routed and processed according to their contents. The engine delivers the messages to their respective destinations, and tracks the status and state of documents. The Orchestration Engine coordinates and schedules message processing, and is used to implement more complicated and long-running processes. [97, p.10] These form the core of BizTalk. Several other components, such as Business Activity Monitoring (BAM), Business Rules Engine (BRE), Health and Activity Tracking and Enterprise Single Sign-on (SSO), are used in addition.

5.1.2 Common enterprise usage of BizTalk

BizTalk has a long history as a hub-and-spoke type EAI solution, but it can be used for various purposes. While it is hard to name one specific scenario above others, there are certain common ways for using BizTalk. In [122], four key areas are mentioned: Enterprise Application Integration (EAI), business to business communications (B2B), Business Process Automation (BPA), and Enterprise Service Bus (ESB) [122, p.68]. Similar scenarios are described in [37]: workflow automation, legacy application integration, trading partner exchange, and organisational message broker [37, p.24]. If such a thing as a typical role for BizTalk exists, it is likely the EAI message broker role (Figure 5.3).

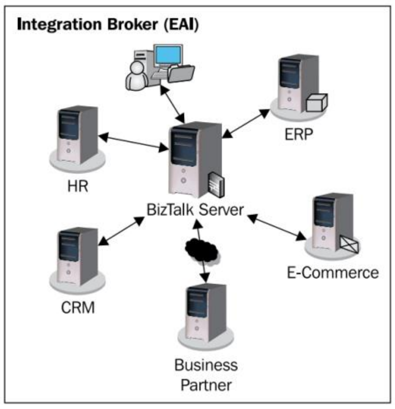


Figure 5.3. *BizTalk Server as an EAI message broker [122].*

BizTalk is used for differing purposes, and within various industries (e.g., in finance, retail, or utilities). It is well suited for moving data between different systems, and it has a well-established position as an EAI tool. However, when discussing ESBs, there are arguments for, as well as against, BizTalk. Using BizTalk as an ESB is covered later in Chapter 5.4.

5.1.3 Technical point of view

From a more technical viewpoint, BizTalk server is a .NET application that is built on top of a set of SQL databases [121, p.15]. (Briefly, .NET is Microsoft’s software framework, and SQL stands for Structured Query Language). BizTalk has certain dependencies, as it requires a few other Microsoft products to support its operation.

A core dependency is that BizTalk installation must have a MS SQL Server as an underlying database system. BizTalk uses internally almost a dozen different SQL databases. Although not strictly required, typically Windows Server is used as the operating system to host the MS SQL and BizTalk Servers. Separate machines for SQL and BizTalk Servers, as well as appropriate redundancy measures, are highly recommended. BizTalk is dependent on MS Active Directory, which provides service account and user access and control [121, p.84]. Thus, a domain installation with a Domain Controller is required (although a test environment can be built on a single Workgroup machine).

It is important to understand that BizTalk is not a working integration solution out-of-the-box. It is a platform on top of which integration solutions can be built. In this sense, BizTalk is much like SQL Server: after installation, it does nothing. Solutions need to be developed on top of the provided platform. [121, pp.13-15] Visual Studio, Microsoft's integrated development environment (IDE), is used for BizTalk solution development. BizTalk installation, solution development, runtime architecture, and administration are discussed in detail in Chapter 5.3.

5.2 BizTalk key concepts and message flow

Key part of understanding how BizTalk works internally is to understand the message flow. This is an overview of how a message enters BizTalk, how it is processed, transformed, and finally routed to its destination. Figure 5.4 below illustrates the path of a message through the key components of BizTalk.

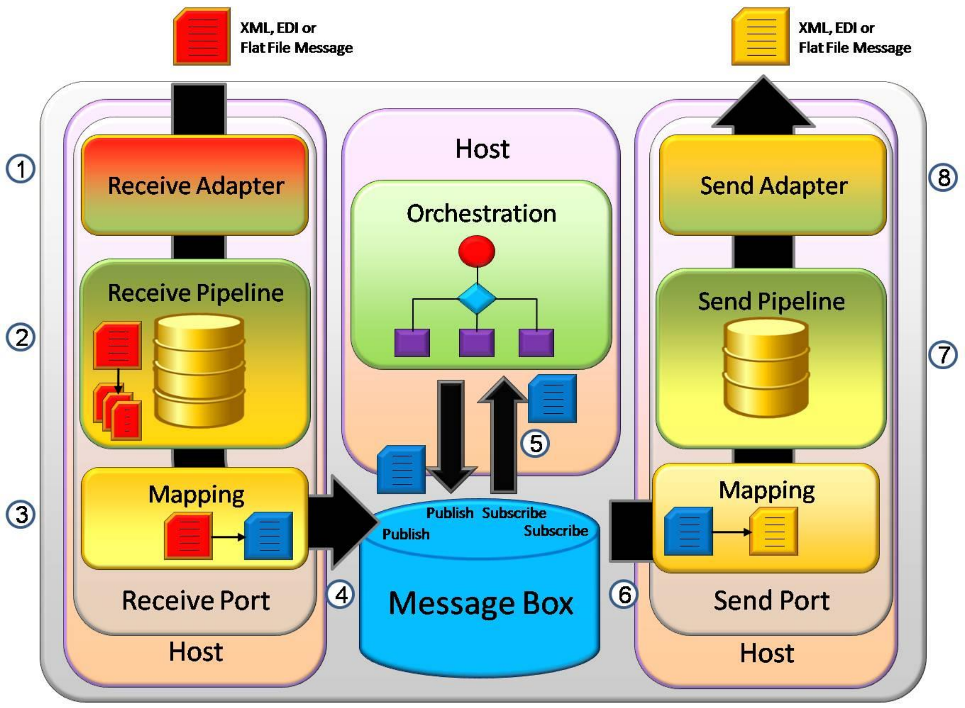


Figure 5.4. *Message flow within BizTalk [97].*

Messages enter BizTalk through receive ports. A receive port is a collection of one or more receive locations that define specific entry points into BizTalk Server. Each location is configured with an adapter and a receive pipeline. Adapters are responsible for the transport and communications part of receiving a message. Receive pipelines can process the message in various ways and prepare it to be published to the MessageBox.

Each receive port is configured with zero or more maps. Mapping means simply transforming a message from one format to another. This is usually done in order to normalise the incoming messages to an internal format. After the message has been transformed into internal BizTalk format, it is ready to enter the MessageBox database. Once the message has entered the database, it is considered "published". The messaging system then checks the existing subscriptions and the message metadata, in order to resolve the orchestrations and/or send ports where the message should be delivered to. The MessageBox is the centre of BizTalk: every message travels through it.

Depending on the solution, a message might be delivered directly to a send port, or to an orchestration. Orchestrations are BizTalk's way of defining and implementing the business process workflow logic. Not all solutions require the use of orchestrations. They are often used when complex and long-running processes are needed, and messaging-only solutions are insufficient.

A message will always leave BizTalk through a send port, whether it was processed by an orchestration or not. The process of sending a message is quite similar as receiving a message - naturally with reversed order of steps. Figure 5.5 below illustrates a messaging-only solution (no orchestrations used), showing the main messaging components within BizTalk. The main difference in send and receive is that there is no "send location".

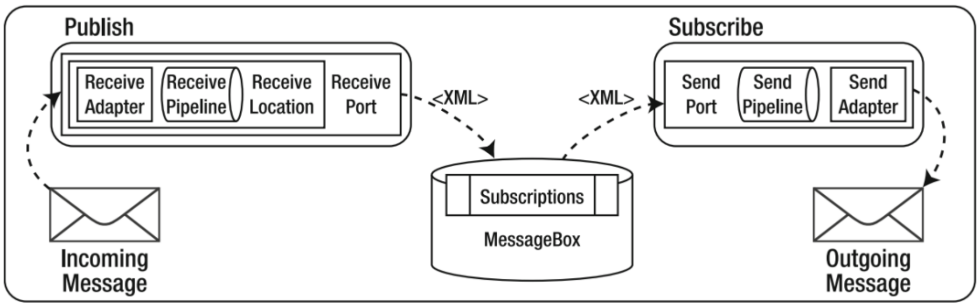


Figure 5.5. *Message flow in a messaging-only solution [13].*

This overview illustrated the fundamentals of BizTalk message flow, the following subchapters providing more details. Much additional functionality such as the Business Activity Monitoring (BAM), Business Rules Engine (BRE), and the ESB Toolkit (ESBT), were omitted altogether. The ESB Toolkit builds on top of these basic messaging and orchestration components (see Chapter 5.4).

5.2.1 Receive ports and receive locations

A message enters BizTalk through a receive port. A receive port consists of one or more receive locations. Figure 5.6 below shows a receive port consisting of two different receive locations (File and Simple Object Access Protocol, SOAP).

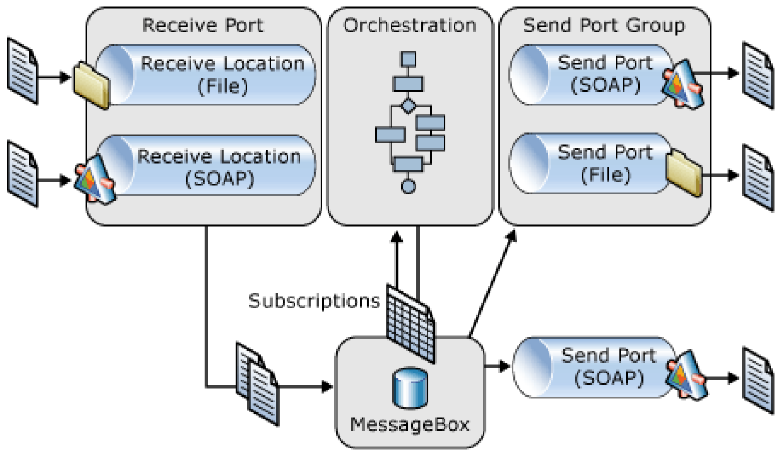


Figure 5.6. *Receive ports and locations [96].*

The idea is that each location represents a single entry point to BizTalk. Each location is configured with an adapter and a receive pipeline, as shown below (Figure 5.7).

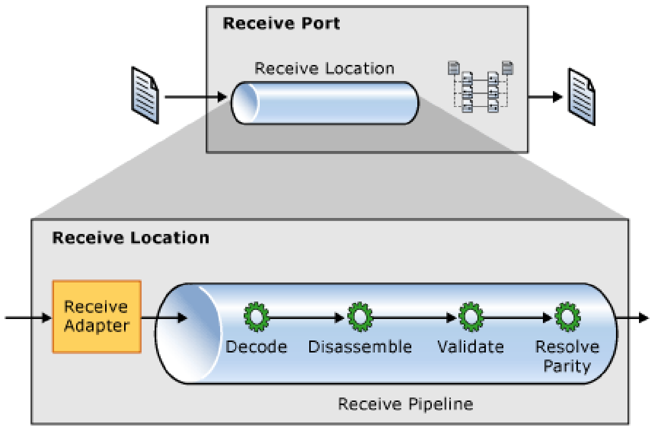


Figure 5.7. *Receive location within a receive port [96].*

Thus, within a port, each location can have a specific adapter and pipeline according to what the incoming message requires for processing. The locations merely serve as logical containers for adapters and pipelines, which are discussed next.

5.2.2 Adapters

Adapters are an essential part of BizTalk, as they provide the points of contact to the outside world [121, p.20]. They handle the communication and transmission of messages, and are the outmost endpoints, providing wire connectivity in and out of BizTalk [87, p.337;122, p.77]. All messages enter BizTalk through an adapter [121, p.20]. In fact, no other component has any knowledge of the endpoints they are dealing with, thus adapters make BizTalk truly loosely coupled [122, p.77].

Adapters can be divided into three classes:

- Transport (or protocol) adapters (e.g. HTTP, POP).
- Line-of-Business adapters (e.g. SAP, Siebel)
- Data (or database) adapters (e.g. SQL, DB2, Oracle) [13, p.335;122, p.77]

BizTalk offers a wide variety of built-in adapters, such as File, FTP, HTTP, POP, SOAP, and WCF (Windows Communication Foundation) adapters. These are called native or integrated adapters, as they are part of the core BizTalk product. They handle the most common communication needs and are not specific to any system or application. In addition, Microsoft offers a library of Line-of-Business (LOB) adapters, which contains adapters for many common enterprise applications (e.g. Oracle PeopleSoft, TIBCO Rendezvous, and IBM WebSphere) [87, pp.337-339]. Third-party adapters are also available on the market. Finally, it is possible to develop custom adapters with the BizTalk Adapter Framework [96].

Figure 5.8 below illustrates an HTTP adapter. In this case, the same adapter can be used for both receiving and sending messages. This depends on the adapter: some only support send or receive functionalities. In general, adapters have different functions and features; some simple, others very complex. [87, p.338]

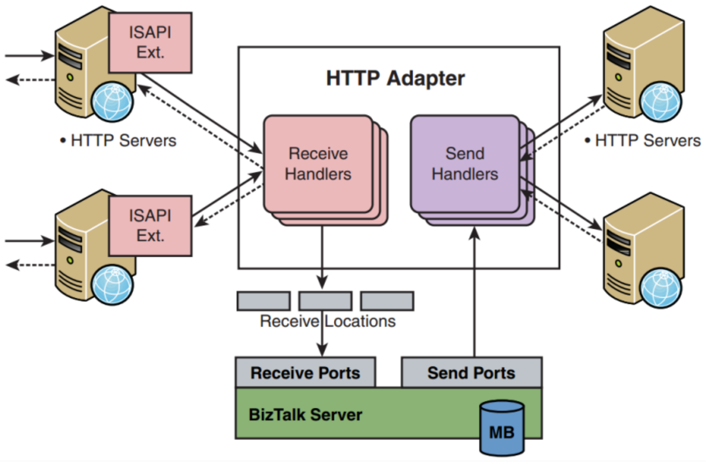


Figure 5.8. *BizTalk HTTP Adapter [87].*

Adapters can support either push or pull models or both. On the receive side, both are common. On the send side, most adapters use the push model. BizTalk explicitly support four specific message interchange patterns: one-way send and one-way receive, request-response, and solicit-response. With custom code, additional patterns and variations can be supported. [87, pp.341-342]

5.2.3 Pipelines and pipeline components

Pipelines are used to normalise data in and out of BizTalk. A pipeline is an implementation of the “Pipes and Filters” integration pattern in BizTalk [121, p.75]. The idea of the pattern is to break down large processing tasks into a sequence of smaller, independent processing steps (Filters) that are connected by channels (Pipes) [61]. A pipeline does exactly that: it is a series of components that are executed in sequence, each providing specific processing to a message [96]. A key benefit of this pattern is that the components are interchangeable, and thus can be re-arranged and used in various combinations. There is less need to change the components themselves. [61]

The receive pipeline prepares the message for publishing into the MessageBox [96]. Pipelines typically perform tasks such as break up inbound documents into separate individual documents, verify or sign documents, process encoded documents, and process flat text files into XML and vice versa. With custom coding, pipelines can be used for a multitude of other things as well. [37, p.116] The receive pipeline has four stages as shown in Figure 5.9.

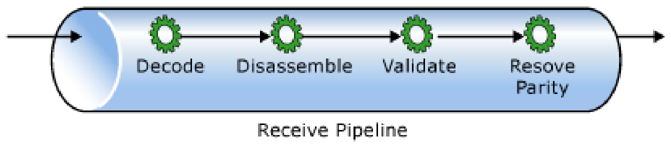


Figure 5.9. *Stages in the receive pipeline [96].*

The send pipeline stages are in reversed order, but otherwise it is almost identical with the receive pipeline. One key difference is that there is no party resolution concept in a send pipeline.

5.2.4 Schemas

BizTalk uses structured documents for all internal messaging and orchestration operations [97, p.15]. Structured messages form the core of most applications. The XML Schema Definition (XSD) language is used to define the structure of messages [96]. Internally, all messages that BizTalk messaging and orchestration engines handle are in XML format [97, p.15].

Schemas are essential for BizTalk for three main reasons. First, a schema defines a message structure which serves as a contract between BizTalk and the system that BizTalk communicates with. If a message received by BizTalk conforms to the schema that both parties have agreed upon, BizTalk can accept it as correct input. Without valid input, BizTalk cannot guarantee valid output. An exact and detailed contract helps in troubleshooting and allows BizTalk to discard invalid input at an early stage. Second, BizTalk creates a message type based on the schemas. This message type is extensively used in subscriptions, where messages can be routed to various locations based on their type. Third, maps (see next subchapter) use schemas as input and output structures when transforming messages. [87, pp.15-16]

BizTalk supports four types of schemas: XML schemas, flat file schemas, envelope schemas and property schemas. Schemas can also be divided into internal and external. An internal schema is essentially a canonical data model used within BizTalk. It decouples the BizTalk internal domain from all the possible external schemas used. Thus, when the external schemas change, only thing that needs to be changed in BizTalk is the mapping. Using internal schemas within BizTalk is a highly recommended best practice. [87, p.19]

The XSD schemas are not specific to BizTalk; they are commonly used in integration. What BizTalk does is it provides graphical tools to help in using them. Although schemas in BizTalk are ultimately represented in XSD, the Visual Studio based BizTalk Editor is used to create, edit, and manage the schemas without having to work with all the intricacies of the XSD syntax. [96;97, p.16]

5.2.5 Maps

Maps enable the transformation and translation of messages. In BizTalk, each port is configured with zero or more maps, helping to convert messages between internal and external types. Maps can also be utilised within orchestrations, when transformations are needed within an internal business process. A map is a graphically illustrated conversion between two XML schemas. It converts an input message that conforms to one schema into an output message that conforms to a different schema [97, pp.16-20].

Maps are created and edited with BizTalk Mapper tool that is integrated to the Visual Studio environment (Figure 5.10). Technically, BizTalk Maps are based on eXtensible Stylesheet Language Transformations (XSLTs). BizTalk maps provide a visual representation of the transformations, and a graphical tool for creating and editing them.

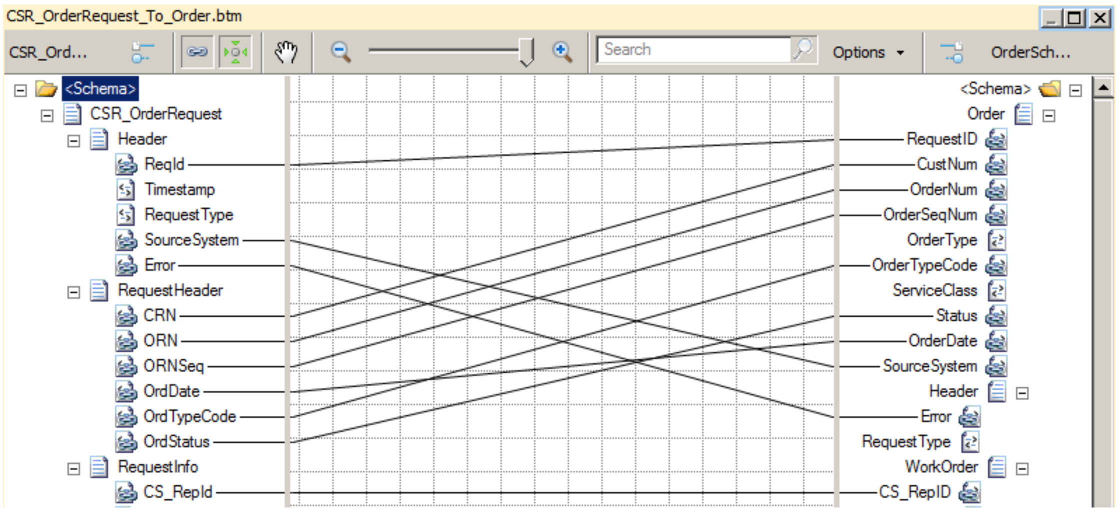


Figure 5.10. *BizTalk Mapper in Visual Studio [97].*

A map defines one-way transformation between schemas by defining conversions between the elements. These conversions can be either simple links that copy values from one element to another, or functoids that perform more complex manipulations on the data [97, p.19]. There are pre-made functoids available, but it is also possible to write new ones as needed.

5.2.6 The messaging infrastructure

The MessageBox database forms the backbone for messaging, and thus, for the entire BizTalk Server product. It is the central database that contains all the in-flight messages that are processed by the BizTalk Server. Often, the MessageBox is thought of as the entire messaging infrastructure, but this is not true. The BizTalk messaging subsystem (the Message Bus) consists of multiple interrelated parts, each performing a specific job. [37, pp.77-78] The MessageBox database is the centre for all action in BizTalk. All messages go through the MessageBox at some point. The operation of the MessageBox is based on the concept of queues [121, p.21].

The term message should be defined in order to better understand messaging. A message in BizTalk is a finite entity that has zero to many parts, one of which is the body part [96]. Messages contain both data and context. Context properties are crucial for routing. It is critical to understand that messages are immutable after publishing. That means that they cannot be changed once they reach the MessageBox. [37, p.82]

Messages are considered published once they enter MessageBox, and the database is then queried for matching subscriptions. When a published message matches an existing subscription, it is then sent to all appropriate subscribers (that is, instances of orchestrations or send ports). After a message is delivered to all subscribers, it is removed from the MessageBox in order to optimise the memory usage and keep the database small and lean. It is important to note that the MessageBox is not used for long-term storage of messages.

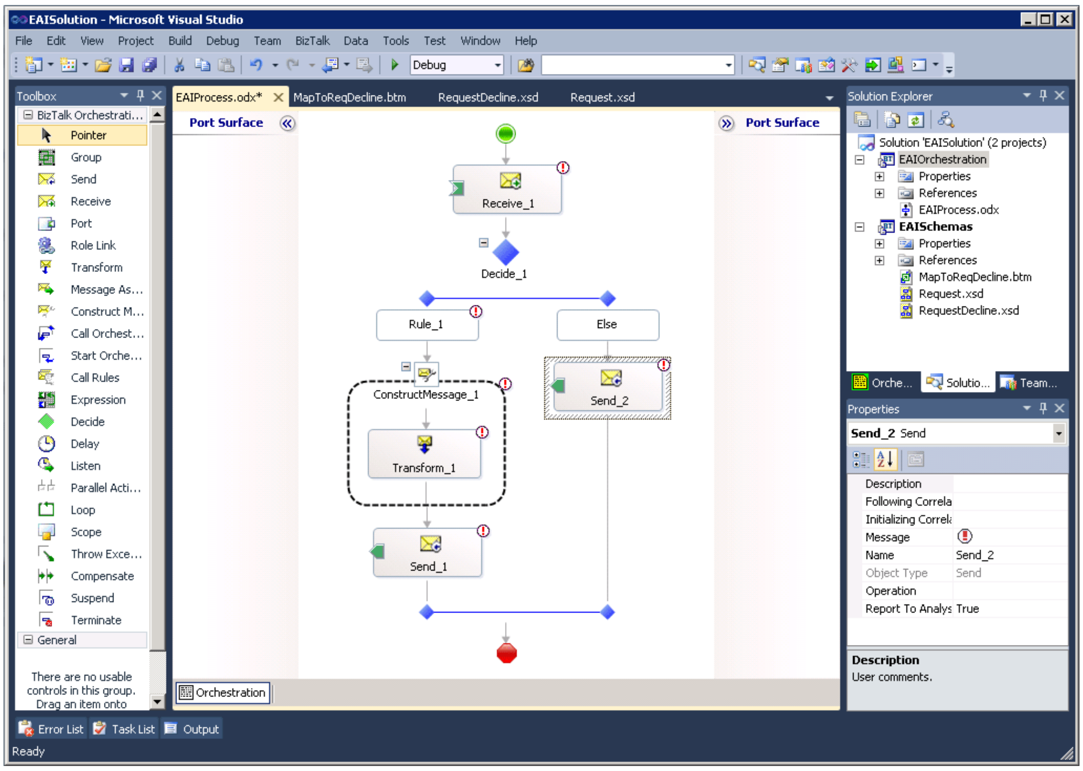
On a technical level, the infrastructure is quite complex. However, a basic introduction is sufficient here, and generally even BizTalk developers and administrators do not need to understand all the specifics. Microsoft’s product documentation and most BizTalk-related books can be referred for details.

5.2.7 Orchestrations

A messaging-only solution that utilises schemas, maps, pipelines, and various artefacts such as ports, is sufficient for many needs. However, BizTalk orchestrations are useful when process or workflow automation or complex routing is required. Orchestrations are useful for managing data flow, decision points, parallelism, exception handling, and other requirements of the interchange between systems [87].

An orchestration is simply a procedural algorithm in a visual form [37, p.272]. They are similar to flowcharts that were used to detail algorithms in functional specifications before sequence diagrams and object-oriented design. An orchestration consists of a series of ordered operations or transactions that implement a business process. [37, p.269] Orchestrations can be nested and they can call other orchestrations, thus making it easier to divide them into manageable-sized parts. This is recommended, as smaller units are easier to manage, and also promote reusability.

Orchestrations are created within Visual Studio in the BizTalk Orchestration Designer, shown in Figure 5.11.



Description	
Following Correla	
Initializing Correl	
Message	
Name	Send_2
Object Type	Send
Operation	
Report To Analys	True

Figure 5.11. *BizTalk Orchestration designer in Visual Studio [96].*

When deployed, orchestrations are compiled into .NET assemblies and installed on the BizTalk databases [37, pp. 7-8]. At runtime, the Orchestration Engine then executes the files created with the designer [97, p.31]. Technically orchestrations are based on the XLANG/s language, which is a sort of a Business Process Execution Language (BPEL), or Microsoft’s “programming in the large” language.

Orchestrations are a powerful tool, because they allow for the rapid development and deployment of complex processes and often require little to no coding [37, pp.7-8]. Partly because of this, orchestrations are often overused in BizTalk solutions. They should be used with consideration, as orchestrations burden the databases with significantly heavier usage than messaging-only solutions. [37, p.272]

5.2.8 Send ports and send port groups

A message should always leave BizTalk through a send port, whether it was processed by an orchestration or not (technically, it may be possible to code orchestrations to directly send messages, but it is likely not the correct approach) [56, p.22]. Send ports operate in a manner very similar to receive ports, although the steps are naturally in reverse order.

Similar to receive side message flow, a map may or may not be applied. Next, the message goes through a pipeline for possible further processing. There is no concept of “send location” within send ports, each port will point to a specific location outside BizTalk. However, send port groups can be used. They are named collections of send ports, and BizTalk uses them to send the same message to multiple locations. A send port may belong to zero or more send port groups, thus having multiple subscriptions.

5.2.9 BizTalk databases

In addition to the MessageBox, BizTalk uses multiple other databases for various purposes. Most important of these is the Management Database, which is a central store for meta-information. It holds all artefacts (e.g. ports, maps, schemas, and orchestrations) that are part of the BizTalk solution. The Tracking database records all events which take place within BizTalk. [121, p.23] It is also the long-term storage for all messages. The Business Rules Engine, Business Activity Monitoring, Enterprise Service Bus Toolkit and Single Sign-On components each use their own database (or possibly multiple ones).

Total of almost a dozen databases may seem excessive, but the use of these highly specialised and optimised databases is essential for making BizTalk a distributed, scalable and fault-tolerant product. Complete listing of databases and their usage can be found for example in: [92;96;121, pp.23-24].

5.3 The lifecycle of a BizTalk integration solution

The BizTalk discussion has so far been centred on the logical aspects of message flow. Transforming a logical, abstract message flow to an actual BizTalk solution running on a server still requires a fair amount of installation, development, and administration work.

BizTalk Server, along with the supporting Microsoft products, provides the development environment, where solutions are created, as well as the runtime environment, where the solutions can be hosted and executed.

5.3.1 Installation

Installing BizTalk, even just for simple testing purposes, has certain prerequisites. Depending on the BizTalk features to be installed, various supporting components and systems must be installed and configurations made. For production environments, careful planning should precede the installation. Microsoft provides thorough instructions and manuals for installation, and it is important to familiarise oneself with all the steps before beginning.

The core dependencies between BizTalk features and supporting platforms and software are illustrated in Figure C.1 (Appendix C). In addition to the basic configuration, examples of typical components to be installed are Internet Information Services (IIS) server role, .NET Framework, Windows SharePoint Services, SharePoint Foundation, and MS Office Excel.

The details are not discussed here. This is just to point out that creating a BizTalk installation that will fit the needs of the enterprise requires careful planning. Instructions are available from Microsoft. For Windows Server 2008 environment, the basic installation, list of supporting software required and the configuration are explained in [94]. Installation with BAM (Business Activity Monitoring) on a multicomputer environment is explained in [93]. Known issues and troubleshooting setup are listed in [98].

5.3.2 Solution development and deployment

BizTalk offers a platform and a working environment to develop, deploy, and host solutions. The specific integration functionality resides within the solutions. Creating BizTalk solutions typically involves the creating of schemas, maps, and orchestrations, followed by a local deployment (for testing). This is done in Visual Studio. Next, the Administration console is used for binding the solution. Creating visibility and monitoring is usually performed with Excel and Tracking Profile Editor (TPE). Testing the (already running) solution is the final step, and can be done in many ways. [121, p.32]

Figure 5.12 shows the main components of a BizTalk solution. The components that are created with Visual Studio are compiled into .NET assemblies. These are then executed at runtime in a BizTalk host instance.

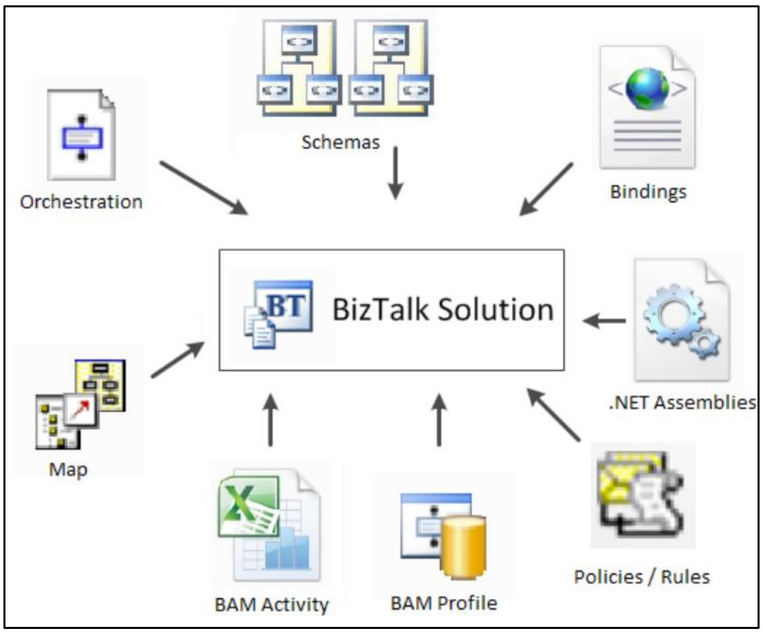


Figure 5.12. *Components of a BizTalk solution [121].*

BizTalk solution development, like any sort of programming, has its guidelines, best practices, naming conventions, and so on. Testing is naturally an important part as well. Further, BizTalk usually offers multiple ways to perform a given task, some more optimal than others. Experience in BizTalk development will help to make the correct decisions. For development guidance, see e.g., Chapter 2 of [121] and [13;37].

During the deployment process 1) the application metadata such as bindings, subscriptions, schemas, and so on, need to be transferred to the Management database, 2) the .NET assemblies that the application comprises of need to be deployed to the servers, and 3) physical endpoints (e.g., file shares, IIS virtual directories, FTP sites) need to be created and configured. Necessary modifications due to changing from development environment to production environment must be made to the physical endpoints. [87, pp.687-688]

5.3.3 Runtime environment

BizTalk runtime environment depends on the specifics of the installation, but each installation shares same basic concepts. BizTalk Server can be installed across multiple physical servers, and the installation consists of various abstract concepts that promote distribution.

The runtime environment comprises various servers (not necessarily physical server machines, but servers in the sense of server roles, such as the IIS). First, there are Application Servers, and these are generally perceived as being the “BizTalk Servers”. Each BizTalk Server can run multiple instances of this role or service. Other important servers or services are Database Servers (hosting the BizTalk databases), Web Servers (used as endpoints for HTTP/Web Services/WCF, and the BAM Portal), and the Enterprise SSO service (provides secure credential storage). These could be hosted on a single physical server, but this is not recommended. [121, pp.25-26]

Key runtime environment components are illustrated in Figure 5.13. The highest level of abstraction is a BizTalk group, which is a logical container for everything in a BizTalk installation [121, p.26]. A group consists of BizTalk runtime machines that share a common Management Database [122, p.74]. Next level of abstraction is hosts. A host defines an abstract, logical runtime container for BizTalk Server resources, such as orchestrations and adapter handlers [122, p.75]. It is presented as a single unit, but can consist of processes on separate physical servers [121, p.27]. Finally, host instances are the actual, deployable and executable runtime processes. A host instance is a physical instance of a logical host, and resides on a single physical machine [122, p.75]. A host instance is where maps, orchestrations, pipelines, and other components all execute [121, p.28].

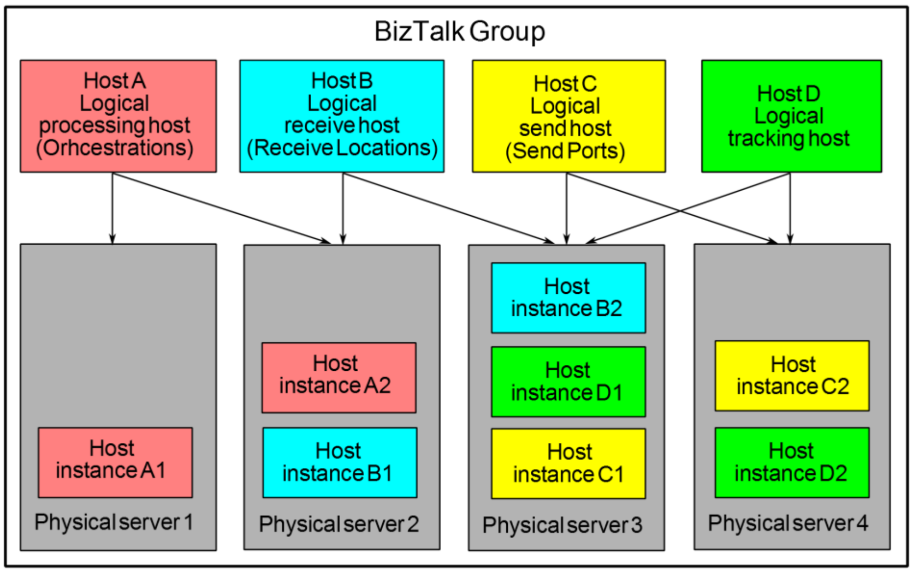


Figure 5.13. *BizTalk Server runtime architecture concepts, adapted from [121;122].*

This somewhat complicated structure improves scalability and availability. First, hosts can contain combinations of artefacts, which helps segregate the duties based on the performance requirements. For example, a single host can be dedicated for orchestrations, if the solution is expected to process large amounts of them. A basic recommendation is to have separate send, receive, orchestration, and tracking hosts [121, p.105]. A single host can just as well contain all types of artefacts (orchestrations, ports, adapters, etc.). Second, a single host can (but does not have to) be deployed as an instance on multiple physical machines. Typically, in a production environment, a host has instances on at least two machines, to provide redundancy and high availability. Yet it is not required for all hosts to have instances on all machines. [122, p.75] These measures allow the distribution of functionality across multiple machines and thus form the basis of the hybrid hub-bus architecture (Figure 5.14).

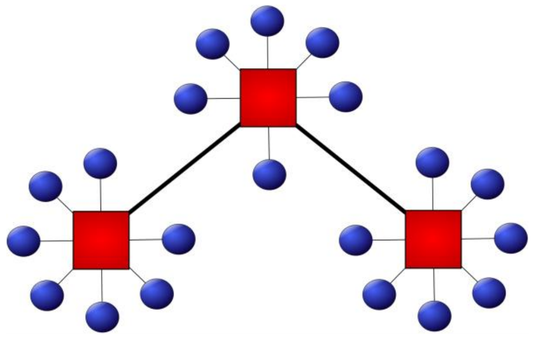


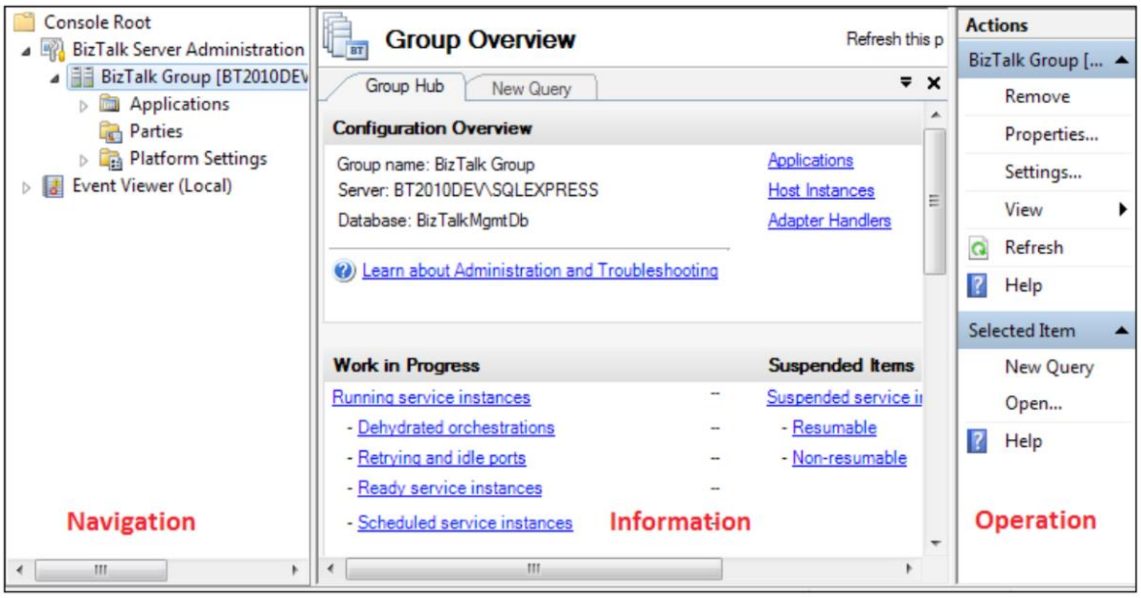
Figure 5.14. *BizTalk hub-bus hybrid architecture, adapted from [87].*

Each machine is shown as a physical hub that shares a centralised message bus, which encompasses the messaging data store (MessageBox), configuration data store (Management Database), and operational and management tools [87, p.6]. The processing capabilities can thus be distributed across different machines. The solution can easily be scaled out by adding servers, or hosts and host instances. Further, different hosts can be assigned with different tasks (e.g., messaging or orchestration). [12;87, p.6]

The hub-bus architecture significantly improves the distribution and scaling options for a BizTalk installation. However, this shows where BizTalk clearly falls short of the ESB criteria. An ESB can be distributed in a fine-grained manner, as the deployment is based on the concept of services. In BizTalk's case, the smallest possible increments (or decrements) are BizTalk servers (and hosts and host instances). Whether this is an actual issue depends on the specific scenario, and is a topic for another discussion.

5.3.4 Administration

Once BizTalk is installed and configured, solutions can be developed in Visual Studio and deployed on the BizTalk Server installation. In addition to these tasks, BizTalk, like any other server, requires administration. There are three main methods for administration work: the Administration Console, a command-line tool, and various scripting or programmability APIs (Application Programming Interfaces) [87, p.669]. The two latter are ideal for routine tasks, but the Administration Console (Figure 5.15) remains the main tool for many configuration, troubleshooting, and management tasks [87, p.669, p.685].



Group name: BizTalk Group
Server: BT2010DEV\SQLEXPRESS
Database: BizTalkMgmtDb

*Figure **5.15.*** *BizTalk administration console [121].*

The Administration Console can in fact be used for developing simple messaging-only solutions without the help of Visual Studio. The console is used for example to tune the BizTalk group, run queries to monitor the current operational state, debug various issues, configure applications, and to evolve the physical topology of the BizTalk group [87, p.685].

5.4 BizTalk as an Enterprise Service Bus

BizTalk has traditionally been positioned as an EAI solution. It was already a well-established product before the term ESB was introduced. As the concepts of SOA and ESB became more popular, EAI was seen more and more outdated as an integration pattern. BizTalk developers were concerned about the future of BizTalk: how Microsoft would position the product in their integration portfolio? Would Microsoft continue to support BizTalk, or deem it obsolete? Throughout the past few years, it has been questioned whether BizTalk is dead. The answer is that BizTalk has not been deprecated, and it is, in fact, at the heart of Microsoft's ESB solution [37, p.645].

Rather than a product, Microsoft sees ESB as an architectural pattern, a set of capabilities that can be provided by a combination of Microsoft technologies. BizTalk Server, with the ESB Toolkit (ESBT), forms the core of this combination. The ESBT started out as "ESB Guidance", a collection of documents and components, which then developed into the first version of the toolkit. ESBT 1.0 was released to be run on top of BizTalk 2006 R2, and ESBT 2.0 was similarly for BT 2009. In BizTalk 2010, the ESBT was upgraded into 2.1, but still offered as an additional feature, requiring separate installation. In BizTalk 2013, the ESBT has become part of the core BizTalk product.

This course of evolution indicates that Microsoft (as well as many others) considers the ESB an important part of modern integration.

The question is: how can a product that has always been an EAI hub, suddenly become an ESB? Is this just about Microsoft trying to repackage old functionality under a new name, in order to extend the lifetime of the product? Historically, many BizTalk solutions have become large, complex, and tightly coupled. That is, they suffer from the common EAI problems. Yet, it does not have to be so, and BizTalk is not inherently inflexible. BizTalk does offer many dynamic capabilities. The main reasons why developers were not using these were that 1) the developers were unaware of the features and 2) a significant amount of custom code is required to make use of the features. [37, pp.646-648]

The ESB Toolkit is trying to bridge this gap between what is possible in theory, and what is achievable with practical amounts of work. It introduces new components and frameworks and provides architectural guidance. In technical terms, it is a codification of many BizTalk best practices. However, it is essentially an abstraction layer, building on top of the existing architecture rather than changing the underlying components. The ESB Toolkit is always used together with BizTalk; it does not function as a standalone installation. The five layers of the ESB Toolkit stack are illustrated in Figure 5.16.

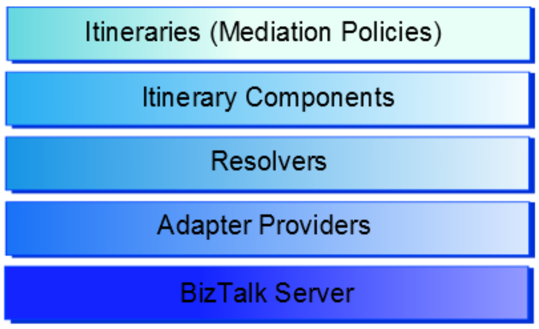


Figure 5.16. *The ESB Toolkit stack.*

The stack illustrates how BizTalk Server forms the foundation for the Toolkit. The ESBT uses the BizTalk mapping engine, adapters, pipelines, orchestrations, rules engine, and the host environment itself. The other layers of the stack contain the components that constitute the actual ESBT (e.g., .NET components, web services, and prebuilt BizTalk components, such as orchestrations and pipelines). [37, p.649]

The two top layers, mediation policies and components, form the basis for implementing itinerary-based routing. The middle layer, resolvers, provides the dynamic, runtime resolution capabilities. Right on top of the BizTalk Server, the adapter providers allow the .NET-based ESBT components and BizTalk Adapters to communicate with each other.

Mediation policies sit on top of the ESBT stack. These policies define how the various mediation components should process a message, and where the components can retrieve the required configuration information. A mediation policy is used as a conceptual term, and a concrete instance is called itinerary (or routing slip). [37, p.659] An itinerary essentially describes a series of steps required to process a message (that is, a series of services to be invoked). This description is then implemented by the mediation (or itinerary) components residing on the subsequent layer.

If not using the ESBT, an orchestration could be used to define the message flow. However, orchestrations are static artefacts that require many operations in order to function (e.g., they must be compiled, registered in the Global Assembly Cache, GAC, and deployed on the ESB). On the other hand, itineraries are nothing but raw XML data, travelling along the message through the ESB. This data is stored in the message context properties, and can be accessed by any component. [37, p.660] If the message flow changes, only thing that needs to be updated is the itinerary: no changes to low-level code or BizTalk, and no need to recompile anything [37, p.663].

The itineraries are technically just XML, so it is possible to manually create them. What BizTalk once again does is it offers helpful tools for this rather cumbersome task. Visual Studio has a specific design surface that is used for itinerary development. The itineraries can then be exported to any environment as XML files, or, more likely, exported directly to the SQL database which functions as a repository. [87, pp.651-652]

The mediation (or itinerary) components perform the actions described by the itineraries. The core mediation components are the generic Routing service and the generic Transformation service (called the ESB services or ESB agents), and on-ramps and off-ramps. The ESB agents are the (BizTalk) components that provide the dynamic routing and transformation capabilities. The on-ramps and off-ramps allow applications on other platforms to leverage the BizTalk-based ESB agents. They provide generic and reusable entry and exit points to and from the ESB. [37, pp.650-652]

One of the ESB keywords is “dynamic”: in order to provide flexibility and reusability, static and hard-coded values should be avoided, and dynamic runtime resolution used instead. However, whether static or dynamic, all the metadata and instructions still need to come from *somewhere.* The resolver framework (the middle layer of the stack) provides the means to dynamically resolve all types of required metadata at runtime, from various data sources. The resolver mechanism can be used to specify itineraries, maps, endpoints, and so on. The ESBT has prebuilt resolvers to support various technologies, e.g., UDDI (Universal Description, Discovery, and Integration), XPath (XML Path Language), and BRE. [50, pp.10-11;87, pp.655-657]

For example, as a message enters the ESB, an itinerary to be used should be specified. This could be done already at client-side, but it is preferred to use server-side specification in combination with an itinerary repository. [87, pp. 651-652] A resolver could be used to perform a UDDI query, execute an XPath statement against the message, or execute a BRE Policy, in order to retrieve the itinerary name. Using the name, the resolver can then retrieve the correct itinerary from the ESBT database and attach it to the message. [50, pp. 11-12] Thus, the itinerary is resolved dynamically, at runtime.

Adapter provider framework basically provides mapping between the ESBT configuration properties and BizTalk adapter properties [50, p.10]. The core problem is that the resolvers use a data format (a dictionary object) that the send ports do not understand. The adapter providers extract the data from this format and place it into BizTalk context properties, which the send port is able to process. [87, p.657] It is worth noting that only those BizTalk adapters that have a corresponding ESBT adapter provider can be used as dynamic off-ramps. The ESBT has multiple built-in adapter providers available, and it is possible to develop custom ones. [87, pp. 658-659]

Figure 5.17 illustrates an example scenario of a message traveling through a BizTalk-based ESB. It shows how the previously introduced components work together to provide the key ESB functionalities. For more information about the ESB Toolkit, and BizTalk as an ESB, see e.g., [50;95].

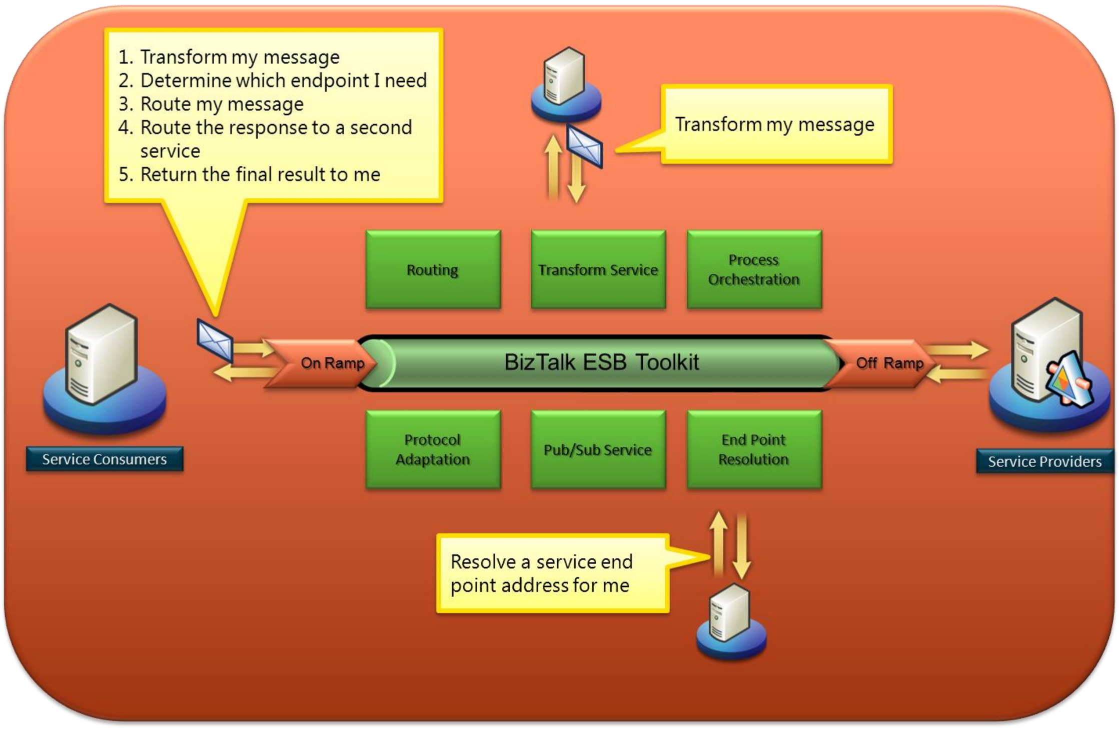


Figure 5.17. *Message flow through the BizTalk ESB [50].*

Whether BizTalk is an ESB or not has been a topic of rather heated debate. Arguably, the most important qualities of an ESB are 1) dynamic operation, i.e. runtime resolution of transformations, routing endpoints, and so on, and 2) internally service-based implementation and deployment. The ESBT can be used to build significantly more dynamic solutions than what is possible with BizTalk alone. The ESBT operation is also internally, up to a degree, service-based. Yet the underlying BizTalk environment, which the ESBT needs in order to function, is clearly monolithic in nature. This is a fact that cannot be easily changed. It can be argued that this disqualifies BizTalk as an ESB. However, rather than entering the debate, it is probably more important to understand whether this limitation has any real effects on the integration task at hand.

5.5 Information security in BizTalk

Planning and building a secure BizTalk architecture, and securing the BizTalk platform, the underlying databases, and operating systems, are all essential to the overall security. An example diagram of a highly distributed BizTalk architecture that takes defence-in-depth into consideration is given in Figure D.1 (Appendix D). Even with a simpler architecture, the security considerations are complicated and must cover various technologies and platforms, e.g., the Windows Server OS, Domain Controller and the AD, BizTalk Server itself, and MS SQL Servers. When planning the environment, things to consider are high availability (redundancy), backups and disaster recoverability, management and tracking capabilities, and so on. A BizTalk environment is complex, and securing it requires knowledge about general information security concepts, quite high levels of expertise with various Microsoft technologies, and significant investments of time and money.

It is not possible to introduce each of the platforms and their security aspects here, and it was never intended in this work. The goal is to analyse integration security in general, rather than to provide platform-specific details. After all, the integration platform used in the project is BizTalk, but it could just as well be some other product.

The general threats to the C-I-A qualities that were discussed earlier are applicable to BizTalk as well. The BizTalk Server 2010 Help offers a STRIDE –model to aid in threat analysis. STRIDE is an acronym from: Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial-of-service (DoS), and Elevation of privileges. Some examples as well as mitigation strategies and methods are given. [96]

On a general level, BizTalk provides all the usual countermeasures for the threats. Encryption (channel- and message level) can be used to prevent information disclosure (confidentiality of data). Digital signatures are used to ensure that data is not tampered with (integrity of data). Denial-of-service threats can be mitigated by only accepting messages from authenticated parties at receive port level (this naturally only protects the MessageBox, it does not protect the network generally from DoS-attacks, and thus does not help if the entire server is unreachable). Further, the size of the received messages can be limited. Various logs and BizTalk tracking capabilities help to (internally) provide accountability. Digital signatures can be used to identify the participants of a message flow. [96]

6 EXAMPLE ARCHITECTURE AND THE DEMONSTRATION ENVIRONMENT

Earlier research indicates that certain concepts, such as service orientation and canonical data model, play a key role in integration. A proposed solution is to develop a service-oriented architecture that uses an ESB as a communication backbone. The IEC Common Information Model is suggested for the canonical data model. In theory, the benefits of these concepts are very clear. The next step is to demonstrate them in practice with proof-of-concept solutions.

This chapter describes first an example for DSO ICT architecture, and then the demonstration environment that was built as a partial implementation of the example architecture. As the integration is implemented gradually, the goal is that more systems can be added to the solution later on.

6.1 Example DSO ICT architecture in SGEM

Figure 6.1 shows the planned DSO ICT architecture in the scope of the entire project. It is essentially a specific instance of the general distribution domain architecture introduced earlier.

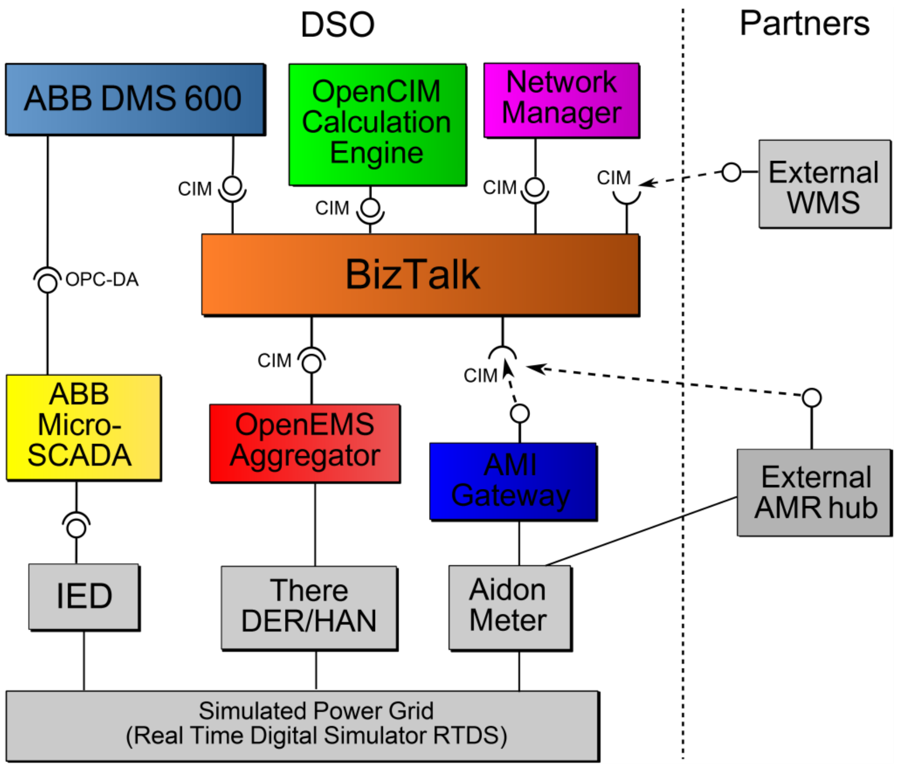


Figure 6.1. *Example DSO ICT architecture in SGEM project.*

In the general DSO architecture, the systems were pictured on a conceptual level (e.g., an ESB, a DMS). Here, each system is a vendor-specific product that is used within the SGEM project (e.g., a BizTalk-based ESB, ABB DMS 600). One of the goals of the whole project is to provide testing environment for integration, and to test real products. The integration platform used here is BizTalk. The following subchapters introduce the various systems connected to BizTalk.

6.1.1 ABB DMS 600 and ABB MicroSCADA

The Distribution Management System used in this project is ABB DMS 600. It is fully functional, mature software system and it is widely used by DSOs in Finland and worldwide. ABB refers to it as a geographical DMS, as it provides geographically based network views in addition to the traditional SCADA functionalities [3]. MicroSCADA, the SCADA product used in the project, is also developed by ABB.

The DMS 600 is deeply integrated to the MicroSCADA, and they have an established communication interface between them (as illustrated in Figure 6.1). However, the DMS 600 can also be used without SCADA, or with other SCADA systems using OPC DA (Open Platform Communications Data Access) interface. [3] In any case, these systems will communicate directly with each other, instead of through the ESB. Possible CIM interfaces and ESB connectivity are not considered in this thesis.

The ABB DMS and MicroSCADA products are complex software systems, and will not be discussed in detail. They have most of the common DMS and SCADA features discussed earlier, and much additional functionality as well. An overview of the DMS can be found, for example, in [77]. For details, refer to the product website [4], and product documentation and manuals (e.g., [2;3;5]).

6.1.2 OpenEMS Aggregator

In terms of the use cases in this work, the key functionality of aggregator is to collect data from customer premises through home automation systems. It then makes this aggregated data available to other systems through the ESB.

The aggregator is based on Nokia Siemens Networks’ Open EMS Suite (OES), an out-of-the-box software product that provides the basic capabilities of an element management system, EMS [106;107]. The home automation system used in SGEM is ThereGate, but it could be another product as well. The aggregator is shortly introduced in source [110] (where an in-depth analysis of the security aspects of the home automation system, and its connection with the aggregator, are also provided). A more thorough introduction is available in [118;119].

6.1.3 OpenCIM Calculation Engine

The OpenCIM calculation engine is provided by InterPSS. It is described as an object-oriented approach to the CIM information model, and as an RDF/XML file processor, rather than an RDF/XML editor. It provides a framework for processing information stored in an RDF/XML file and transferring it into other applications. [67] OpenCIM uses Java and it is built on standard-based Eclipse technologies [65;67].

According to InterPSS, the OpenCIM goals were to provide a tool that does not require high-end hardware, complicated configuration, or enterprise relational database software. This is necessary for keeping the software cost-effective and affordable for utilities of all sizes. Further, it should be possible to adapt or extend the OpenCIM to accommodate enhancement and customer-specific requirements, and integration with other power system applications should be easy. [65;68] Whether these goals were achieved is not analysed in this work. The OpenCIM provides much functionality, but here the only requirement is to perform load flow calculations based on a CIM network model (for the purposes of the use case A, see Chapter 7.1). Thus, the OpenCIM does not require further introduction. Detailed information is available in, e.g., [65;66;68].

6.1.4 Cybersoft Network Manager

Network Manager is a solution used to support the traditional Network Information System functions. It offers tools and functionality for better and more efficient operation and planning of the network. This description is based on the information available at Cybersoft website [31]. The software is currently not available for testing purposes in this project. It is mentioned here, as it is part of use case A (see Chapter 7.1).

The Cybersoft Network Manager is a collection of browser-based solutions for asset management and operations for both distribution and transmission domain. According to the company website, the solution is modular and flexible, and based on standards and openness. The Network Manager is based on IEC CIM standard, it is easy to integrate to other systems, and it is also available as a service. Some key benefits listed are: increased reliability of the network, increased transmission capacity, decreased investments, improved safety, optimized workflow, and lower overall costs. [31]

6.1.5 Other systems

In addition to the systems already introduced, Figure 6.1 shows a few others, such as workforce management systems (WMSs), and AMI (or AMR) gateway. A WMS is used to manage and instruct the crews that handle, for example, repairing and installation tasks on the field. Depending on the product, the functionality could also be part of the DMS. The WMS is also referred to as Field Force Management System (FFMS). In this work, the terms WMS and FFMS are used interchangeably. The AMI gateway handles communications with the smart meters.

There could be any number of various smaller software products in use at a DSO, such as the Coordinated Voltage Controller and State estimator described later in use case C (see Chapter 7.3), and a specific network topology conversion service. These systems are not available in the demonstration environment, and will not be described here.

6.2 Demonstration environment

One of the key requirements for ESB-type integration is that it should be easy to build the solution incrementally. The demonstration environment (Figure 6.2) is a partial implementation of the proposed architecture, and the parts that were implemented are described here. The goal of this work was to provide a working BizTalk installation in the AIN (ain.rd.tut.fi) network. Various other systems in the ELE (ele.tut.fi) network could then be integrated using the BizTalk installation.

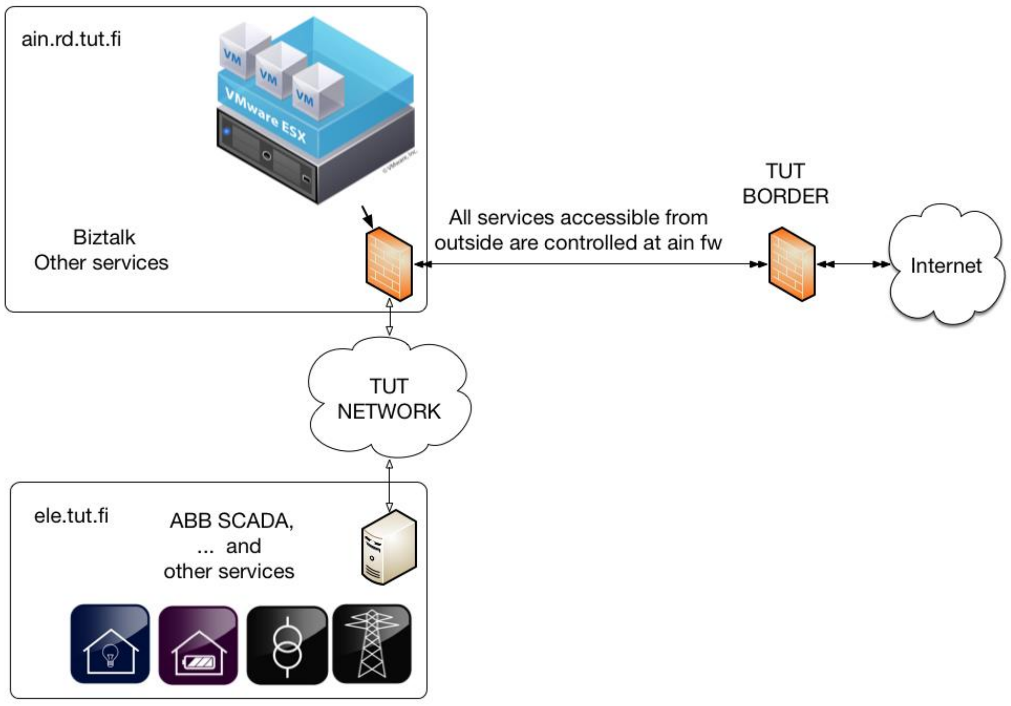

ain.rd.tut.fi
VM
VM
VM
VMware ESX
Biztalk
Other services
All services accessible from outside are controlled at ain fw
TUT BORDER
Internet
TUT NETWORK
ele.tut.fi
ABB SCADA, ... and other services

Figure 6.2. *Demonstration environment.*

The BizTalk environment can be used as a starting point for further integration. The environment is virtualised (see next subchapter) and it is easy to add systems into it. For this reason, the entire implementation of the demonstration environment currently resides at the AIN network. However, for future testing, the DSO ICT systems (such as DMS, SCADA, and OpenCIM) should be installed in the ELE network. That would be a more realistic installation, when all the systems are not on the same physical server.

6.2.1 Virtualisation environment and tools

Even a simple BizTalk testing environment can grow into complex multi-computer installation. Virtualising the environment offers many benefits: it is simple to add new machines, reconfigure the environment, or go back to an earlier state within a virtual machine (so-called snapshot). These are particularly useful things in test environments, where reconfigurations and changes are often made, and the system easily corrupts.

Basic idea of virtualisation is that a single physical machine can run multiple virtual machines, as illustrated in Figure 6.3. A layer called hypervisor, or Virtual Machine

Manager (VMM), is added between hardware and operating system. The additional layer naturally does reduce performance, which should always be taken into account when planning to virtualise an environment. Security is also important in virtualisation: should the hypervisor be compromised, all virtual machines hosted within it could be compromised.

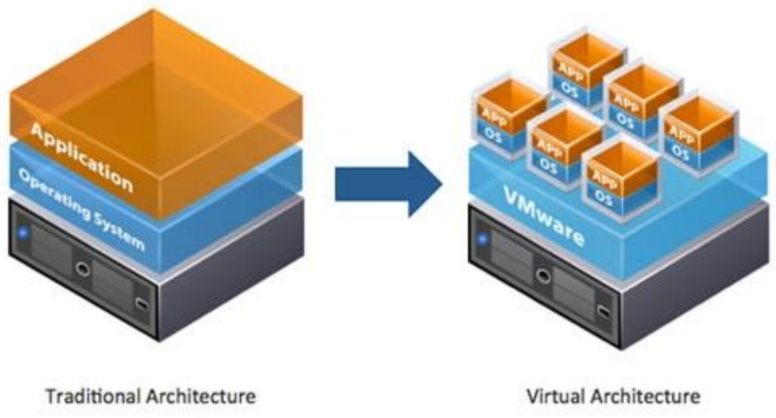


Figure 6.3. *The concept of virtualisation [144].*

The BizTalk environment was built on top of a VMware ESXi virtualisation platform, provided by the Automation and Information Networks (AIN) research group. ESXi is a Type 1 hypervisor; often called native or bare metal hypervisor, because it runs directly on the host hardware. Type 2 (hosted) hypervisor runs as a program within an OS, an option which is not discussed here. For detailed information about virtualisation with VMware tools and products, see, e.g., [139;143].

6.2.2 Configuration of the environment

Figure 6.4 shows a more detailed view of the environment that was presented in Figure 6.2. The components in the AIN network, that is, the virtualised BizTalk environment and the management server, are installed as shown. In Figure 6.4, the DMS, OpenCIM, and other DSO systems are pictured in the ELE network. However, these are currently installed in the ESXi platform along with BizTalk. Future development need is to move the DSO systems from the AIN servers to other machines. These could be located in the ELE network, or they could reside somewhere else entirely (e.g., public cloud-based services accessible in the internet).

The AIN Server A has VMware ESXi 5.0 hypervisor platform installed and it hosts the BizTalk environment. The ESXi itself offers only the runtime environment for the hosted virtual machines. The Server B is used for managing and configuring the virtual environment and administering the virtual machines within Server A. VMware vSphere is installed on a virtual machine running on the Server B. The vSphere software is used for e.g., creating new virtual machines and configuring the virtual networks within the ESXi platform. It is recommended to separate the management traffic and the traffic of the actual virtual machine operations, thus, two separate physical network interface cards (NICs) are used.

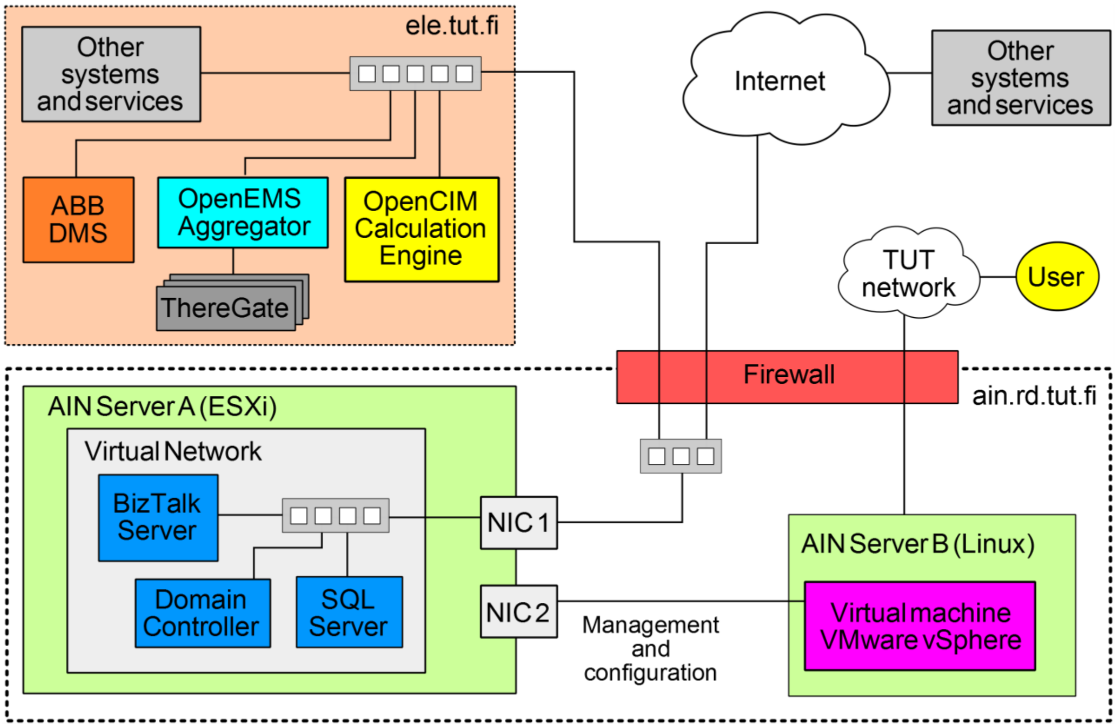


Figure 6.4. *Detailed view of the demonstration environment.*

The benefits of this virtualisation configuration are that it provides a securely accessible platform that can be easily configured and changed, and has sufficient performance.

6.2.3 BizTalk 2010 environment

A test installation of BizTalk 2010, illustrated in Figure 6.5, had been performed and was in place when starting this work. This is a Windows Workgroup environment. The BizTalk Server 2010 is installed on a virtual machine running Windows Server 2008 R2 operating system. For simplicity, the MS SQL Server is installed on the same machine. This can be done when testing BizTalk, but it is not recommended in a production environment. The free test version, SQL Server 2008 Express Edition is used here.

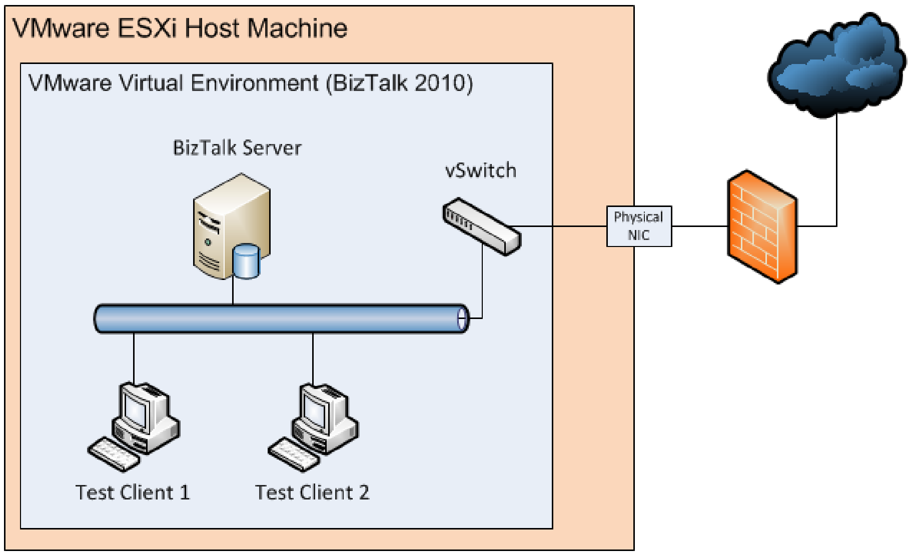


Figure 6.5. *BizTalk 2010 demonstration environment.*

To perform initial testing for BizTalk solutions, two virtual machines with Windows 7 were installed. These test clients had no specific software running; only purpose was to test that data could be transmitted between machines. To gain understanding of how BizTalk works, example scenarios from Microsoft's BizTalk material and various BizTalk books were implemented. These were not Smart Grid –related integrations.

One goal was to test BizTalk's ESB capabilities, but installing the ESB Toolkit into this environment proved to be problematic. It would have required a change into a domain environment. This would have altered the name(s) of the machine(s), causing problems with the SQL Server, and thus with BizTalk. With virtualisation, it proved to be simpler to build an entirely new environment, and the BizTalk 2010 environment could also coexist with the new solution. Another reason to perform a new installation was the release of the new BizTalk Server 2013 Beta.

6.2.4 BizTalk 2013 beta environment

Microsoft launched new versions (the "2012" line) for its core server products, its database servers, and Visual Studio development environment. A beta version of the new BizTalk was also released. In Microsoft's naming convention, new major versions are named according a year (e.g., Windows Server 2008). Smaller, yet still significant updates add a revision number (e.g., Windows Server 2008 R2). This indicates that the release of the new "2012" line was important. Before the beta release, the new BizTalk was unofficially referred to as "2010 R2". The fact that it was eventually named "2013" indicates that it has significantly changed.

The BizTalk 2013 beta environment was built from the ground up, and the result is pictured below in Figure 6.6. This is a Windows Domain environment.

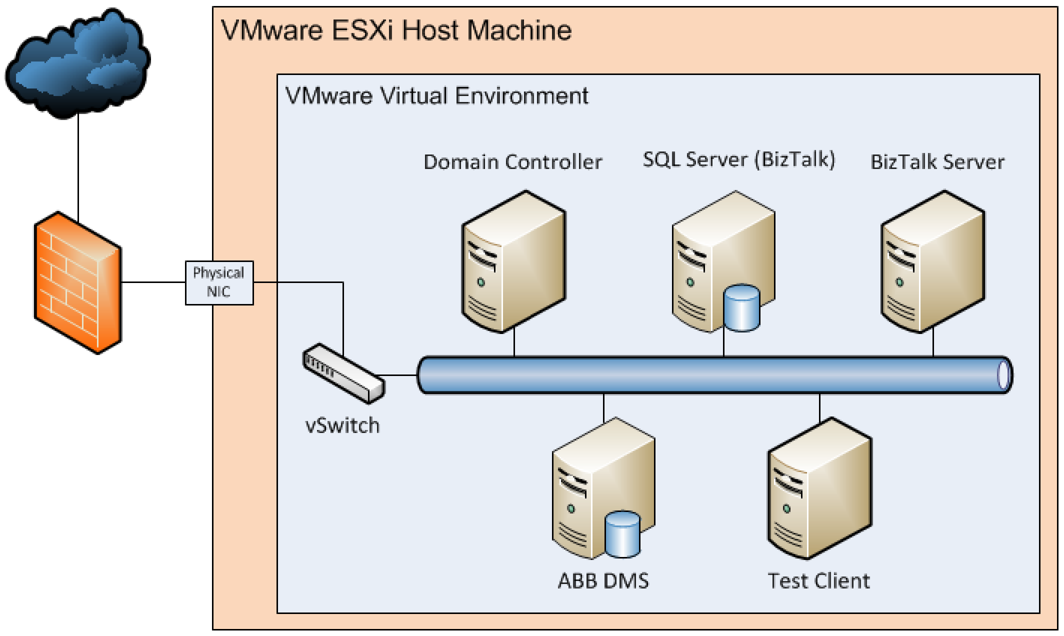


Figure 6.6. *BizTalk 2013 demonstration environment.*

The BizTalk environment is formed by the Domain Controller (DC), the SQL Server, and the BizTalk Server itself. These are all running Windows Server 2012. As a recommended best practice, the DC acts only as a domain controller and has no additional roles. The SQL Server is installed on one (virtual) machine, and the BizTalk Server on another. This kind of installation is sort of a minimum realistic configuration for BizTalk (without any redundancy measures).

The ABB DMS and Test client machines shown in Figure 6.6 are not required for BizTalk, they are specific to this testing environment. Originally, the plan was that the AIN environment only hosts the BizTalk installation. All the systems to be integrated should reside on other environments. However, no outside systems were available during this work. In order to test the BizTalk, the ABB DMS and Test client machines were created within the virtualisation environment. For future work, these should be installed within the ELE laboratory.

The DMS 600 is installed on a Windows Server 2008 R2 virtual machine. It also requires a SQL database, and in this case MS SQL Server 2008 R2 SP1 Express Edition is installed on the same virtual machine. It is possible to export the network model from the DMS database and produce a CIM-formatted XML file. However, this functionality was not officially part of the DMS. It was offered as an additional tool that is still under development.

The test client is a Windows 7 machine that has the OpenCIM calculation engine installed. This client is used for testing file transfers between different machines, and it represents the OpenCIM system. Other software can be installed on this machine when necessary. It is also possible to add more client machines to the environment, if and when needed.

7 SGEM SMART GRID USE CASE EXAMPLES

Three example use cases served as the starting point for demonstrating the integration in practice: network model exchange, fault repairing, and active voltage control. The cases are based on the SGEM research paper "Examples of inter-application communications in a DSO" [89].

The use case starts with a textual description as given in [89]. The analysis starts with the creation of data flow diagrams (DFD) for the purpose of analysing the information exchange between the systems and the information security requirements for each use case. Most of the participating systems listed in the use cases are not yet part of the demonstration environment, and the ones that are do not yet have (fully functional) CIM interfaces. Thus, implementation is possible only in a very limited way; however, implementing is discussed and guidance for further development is provided.

The detailed analysis and the implementation efforts of the use cases clearly show that there are still challenges to solve. Defining and implementing use cases is an iterative process, and for the cases discussed here, the work had just begun. The key issues are collected under the heading "lessons learned" for each use case.

7.1 Case A: Network model exchange

The first use case is network model exchange, where the static network model is exchanged between DMS and an external load flow calculation engine. A network manager operates as a proxy between the two systems. This case is based mainly on the IEC 61970 profiles (61970-452 and 456), unlike the two other cases. The data exchange is described as follows:

1. DMS exports the network model to Network Manager.
2. Network Manager exports the network model to OpenCIM.
3. OpenCIM calculates load flow and exports the results to Network Manager.
4. Network Manager forwards the load flow analysis result to DMS. [89]

In another paper by the same authors, the use case is described as follows: "The first use case is to integrate ABB DMS 600 with a calculation engine based on OpenCIM: DMS exports the whole network model and some measurements to the calculation engine; the calculation engine then performs load flow calculation, and exports the result to DMS." [88]

The authors observe an issue in the use case, namely, how to model the load flow calculation result in the CIM format. This is required for exporting them to other applications. The paper suggests two solutions. One is to choose an existing standardised profile (e.g., ENTSO-E model exchange profile or IEC 61970-456), the other is to define a context-specific CIM profile, possibly by taking an existing profile as a starting point and extending it for this particular use case. [88] This issue remains unsolved.

7.1.1 Data flows

The network model exchange use case was chosen as the starting point for the integration analysis, as it only has a few endpoints and data flows. The only data moving in the use case are the network model and the calculation results, as illustrated by the DFD in Figure 7.1. The unsolved problems related to the use of CIM are irrelevant for this data flow analysis.

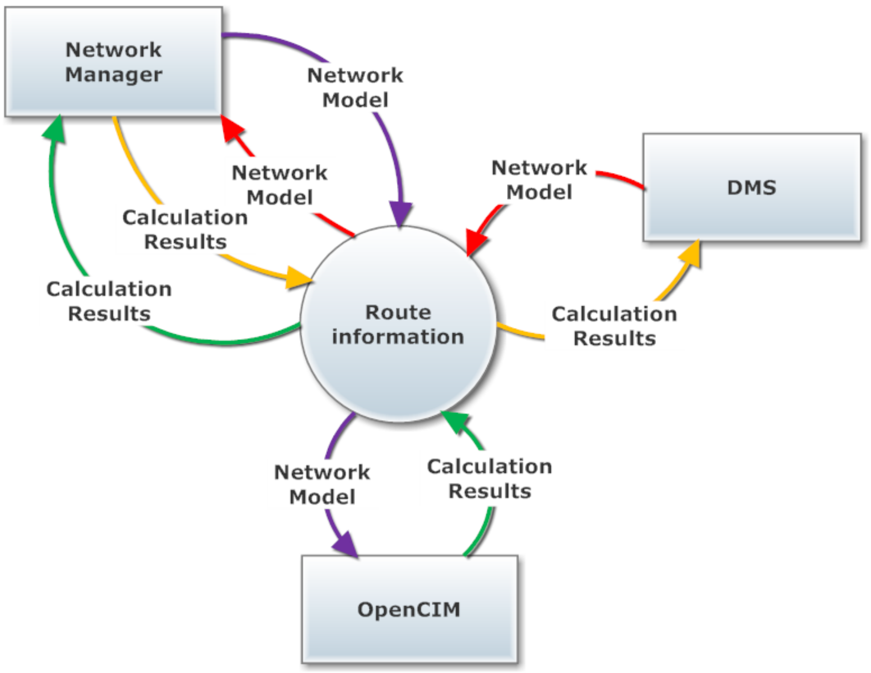


Figure 7.1. *DFD for the network model exchange use case.*

A core idea of the pub-sub messaging pattern is that the publisher does not need to be aware of the subscribers; it only publishes data to the message hub. For example, the OpenCIM publishes the calculation results, but from its viewpoint it makes no difference whether there are subscribers or not. Either the DMS or network manager, or both, may subscribe to that information (and for future use cases, other systems as well).

If the network manager will not actually process the data prior to sending it to the DMS, there is no reason to use it “as a proxy”, as this adds no value. The network manager may update its internal network model, and may also have need for the calculation results. However, if it does not publish any results of its internal processing, it is only a data sink. Thus, in this use case, it can subscribe to the data, but that does not affect the operation of the DMS or OpenCIM. The DFD in Figure 7.2 illustrates a simplified version of the use case where the network manager is subscribed to both the network model and calculation results, but does not alter them in any way.

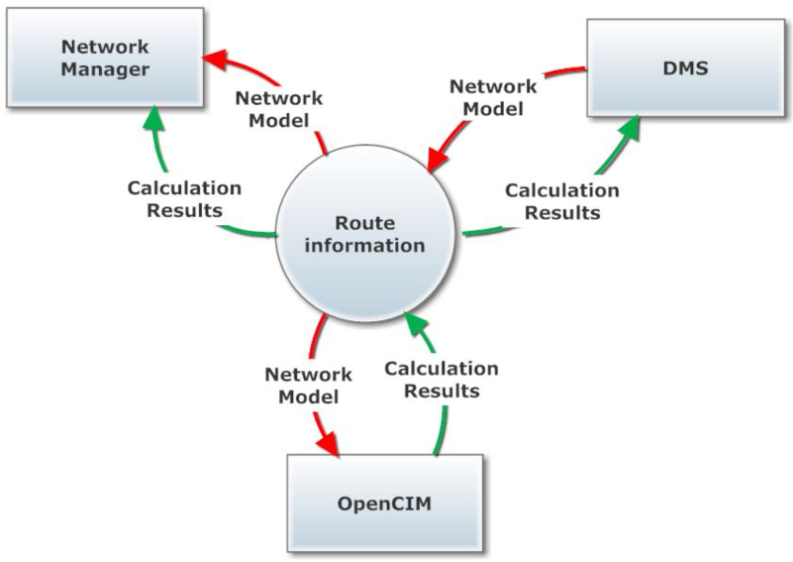


Figure 7.2. *Simplified DFD for the network model exchange use case.*

If the network manager processes neither the network model nor the calculation results, it can be entirely removed from the use case. This does not affect the operation of the DMS or OpenCIM. The further simplified use case is illustrated in Figure 7.3.

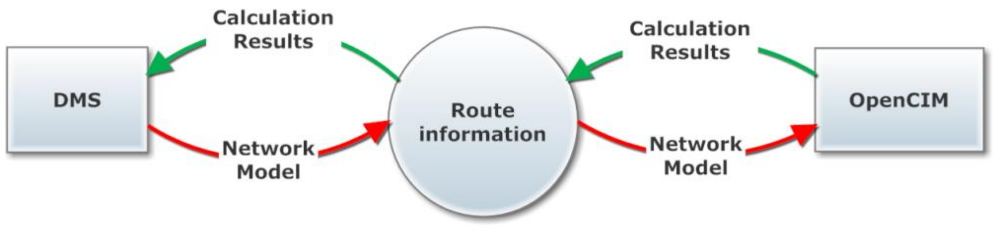


Figure 7.3. *Simple DFD for network model exchange use case (w/o Network Manager).*

This further simplified use case is applied in the implementation. This is necessary as the network manager software is not currently available for testing. However, it is one of the benefits of a dynamic integration solution (and the pub-sub messaging pattern) that it is easy to later add the network manager to the solution. If the network manager does have a role other than a mere data sink, however, then the entire use case needs to be implemented as it was illustrated in the first DFD (Figure 7.1).

7.1.2 Information security

The goal is to analyse the information security requirements of each data flow in a use case. This use case consists of only two data flows: the network model and the calculation results.

A starting point for security analysis is to define the information content of the flows (e.g., the network model or calculation results). Then, the analysis is about asking questions such as: Is this information confidential? Who might benefit from access to it? What happens if the data is wrong or incorrect? Is there a way to know if the data received is incorrect? How can the sender/receiver be authenticated? What if this data is not available 1) in a given timeframe or 2) at all? These are just a few examples of relevant questions. There are plenty of well-established methods and sample question lists for a risk analysis or threat model analysis (TMA). Naturally, these need to be applied to the domain, the environment, and the specific case at hand. This requires collaboration between information security experts, who have the knowledge and mind-set to ask the right questions, and domain experts, who understand how the system works and what it is intended to do, and thus have the expertise to answer the questions.

Table 7.1 provides an example of how the contents of a data flow could be analysed from the information security point-of-view. The example may not be comprehensive.

Table 7.1. *Example data flow security analysis.*

Information content: **Network model**	**Confidentiality**	**Integrity**	**Availability**
Priority/value	Low	High	Medium
To whom is this valuable? (Owner, user, systems, attackers, etc.)	The confidentiality is not critical. The model changes over time, and even if the confidentiality is compromised, the data acquired is soon invalid.	Integrity is highly valuable for anyone who will use this	Availability is generally important, but the network operator participates in this use case, so real-time aspects are “human-time”.
What are possible consequences if compromised	Unlikely mission-critical if confidentiality is compromised.	Consequences can be severe. For example, all the calculation results based on the model are incorrect, if integrity is compromised.	Depends on the timeframe. Immediate availability is not critical. Extended periods of unavailability will start to cause problems in network operation.
Who might attack, how, why?	Attacks against confidentiality possible, but unlikely the most significant issue.	Malicious party who wants to disrupt the operation might target this. Attacks involving an insider may easily affect integrity. Non-malicious aspects probably even more significant issue.	A denial-of-service attack is possible; would probably be targeted more generally against the system. There are likely use cases with more demanding real-time requirements.
Unintended, non-malicious aspects that can cause problems (errors etc.).	Misconfigured systems or human errors could lead to loss of confidentiality.	Error in transmission, human error, outdated data etc. more likely an issue. The data could be erroneous to begin with (before the transmission begins), but this is not an information security issue.	Many types of system failures might affect this (e.g., the service is unavailable, there are problems with the DMS, or with the database that stores the network model).
Countermeasures, how can this be protected (on a general level).	Channel-level security can be used during transmission. Generally, encryption unlikely necessary.	Checksums, timestamps, etc. to prove integrity and correctness (to a degree).	Redundancy at some level. Real-time performance is not critical, so using e.g. resending to assure successful transmission is applicable.

While this kind of an analysis for each data flow is important, but in the end of the day, information security is a money matter. There is always the cost-benefit ratio to consider. For example, if one data flow from a certain system needs to be confidential but others need not, it may be simpler to encrypt all data flows (channel-level) rather than analyse them in detail. Or, building of a redundant transmission channel that will ensure data availability in case another channel is unavailable would provide improved availability for all the flows between the systems, even though only some would require it. In general, the analysis can be simplified by considering security measures that have a more widespread coverage than a single data flow (e.g., channel-level security methods instead of message-level, whenever appropriate).

A crucial aspect of the analysis is to define the security (trust) boundaries. Even though the current environment may be mostly (or even fully) under the direct control of the DSO, it is unlikely to remain so in the future. Building security based on the assumption that data remains within an organisation’s control will probably lead to issues later on. It is safe to assume that, in the future, most data flows may cross organisational borders, due to the increased use of service outsourcing and cloud-based solutions. In SOA, the organisational borders are unclear or even non-existent. This makes it much harder to implement information security, as the content may be processed and transmitted in very different environments.

In addition to this detailed (flow-by-flow) method of analysis, it is important to understand the bigger picture. In general, what happens if this use case cannot be implemented or there are information security related issues? It is hard to analyse this without understanding, in general, what is done in the use case and why. This indicates that a more abstract, conceptual model is needed.

7.1.3 Implementation with BizTalk

The use case could be implemented in various ways using BizTalk. The following extremely simple scenario is provided as a starting point. The DMS can export a network model in CIM format, but this functionality is not yet officially part of the DMS, it is offered as a custom tool that is under development. Currently, there is no interface that would make the model available for other systems or users. The custom tool outputs an XML file generated on the basis of the network model data stored in the DMS database.

Some BizTalk functionality can be demonstrated with a very simple file transfer based messaging solution. The solution is, in fact, so simple that it can be entirely constructed with the administration console; it is not necessary to use Visual Studio.

The file transfer is implemented as follows: BizTalk picks up the network model (XML file) from a shared folder on the DMS machine. A pass-through pipeline is used, meaning that the XML file is not processed in any way. No mapping is used either, the network model is a payload in the message. It is technically possible to map the file into an internal BizTalk format, but, in this case, it would make no sense. Internally, BizTalk does not use the network model for any purpose. Moreover, the XML file is very large, with an arbitrary number of fields. Thus, it is more reasonable for BizTalk to handle it as a payload, rather than process it in any way.

The routing is based on the information about the receive port. The send port that sends the network model to the calculation engine simply subscribes to messages received from a given receive port. When a network model arrives at the receive port, it is routed to the send port that sends it to OpenCIM. Now, if the model should additionally be sent to, say, the Network Manager, it only requires the addition of another send port that subscribes to this same receive port. This is a really simple configuration task.

Returning the load flow results to DMS is still a question mark. The issue is twofold: 1) how to model the results using the CIM and 2) how to export the results from the OpenCIM for other systems to use. This clearly shows that there are still issues in the use of the CIM.

Even this simple scenario shows some of BizTalk's capabilities. Building this kind of a solution is easy, and BizTalk automatically provides reliability, error and exception handling, tracking, and so on. This type of message exchange between various endpoints is exactly what BizTalk was originally designed for, and it does this very well. Yet, it has very little to do with using an ESB or being service-oriented. The solution is essentially a point-to-point type of link between the two endpoints (even though here implemented using a hub). It does not embrace the key ideas of service orientation or promote such goals as reusability and service composition.

7.1.4 Lessons learned

Clearly, the implementation efforts did not lead to an ESB-based SOA, and the accomplished implementation remains very limited. Many issues were recognised while analysing and implementing the use case. First, the data flow analysis showed there were some issues with the pub-sub messaging pattern. Second, the information security analysis brought up the need for a more abstract model that states in general terms what the use case is all about. Finally, the implementation phase showed that the way the use case is currently defined does not necessarily lead to a service-oriented solution. An ESB is not required for such an implementation.

Currently, the use case description starts at a fairly technical level. The whole purpose of exchanging a network model and performing load flow calculations is probably clear to a person with a background in electrical engineering. For an information security or integration professional, *what* is done (on a conceptual level) and *why* it is done, is likely to remain vague. Building secure integration within the Smart Grid requires expertise from many fields. Everyone participating should have an accurate idea of what a use case is all about, why it is done, what benefit it bears, and what happens if it, for some reason, goes wrong.

Among the first steps, an essential model of the use case should be developed. It explains the use case in general terms, defining what is done without going into technical details. Figure 7.4 shows a DFD illustrating an essential model for network model exchange.

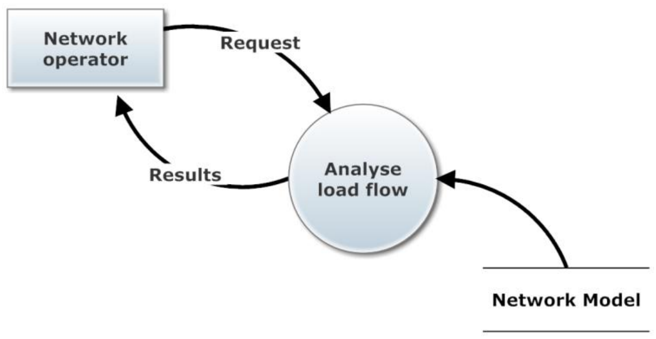


Figure 7.4. *Essential model for the network model exchange use case.*

The essential model shows that the network operator initiates the case by requesting a load flow analysis. The network model is required to perform the analysis, and the results are then returned to the network operator. A textual description should be added to further explain, for example, why this is done, how often it would happen, consequences if it cannot be done (at all, or within a timeframe), what if the results are erroneous, and so on.

In order to move towards service-orientation, certain services are obviously needed. A good example is a service that would perform the load flow calculation, based on a network model given as an input. The DMS could invoke this service, and provide the CIM formatted network model as an input. The load flow analysis service would then return the calculation results to the caller (the DMS). In this case, the transaction would be initiated and orchestrated by the DMS.

Making the network model available for other users is another prime example of a possible service. Both of these services deal with rather complex things, so it might be necessary to have a few different variants of these services (for example, a service that provides the entire network model, and another that provides the changes done after a specified time).

This is a starting point, but merely wrapping a few functionalities to be offered as services does not make the architecture service-oriented. Similarly, connecting these services using BizTalk does not yet constitute an ESB.

An ESB-based, service-oriented solution would operate differently. The request to perform load flow analysis enters the ESB. The request can be initiated by the network operator, happen automatically as a part of some other use case, or it can be time-based, for example. The ESB would dynamically solve which itinerary to use, and attach it to the request, in order to invoke the right services. First, a service that provides the network model is invoked. The next step is to send the original request and the network model to a service that performs the load flow analysis. Finally, the results of the analysis are routed back to the original requestor. Optionally, the load flow analysis service could invoke the network model service and request for the model, if it was not attached to the input. In the original description of the use case, pub-sub messaging was mentioned. This sort of itinerary based scenario does not require (or even support) the use of pub-sub messaging.

7.2 Case B: Fault repairing

The second use case concerns fault repairing. The data exchange is described as follows:

1. Smart meter captures a problem at a customer's residence. The Automatic Meter Reading (AMR) gateway raises an alarm, which is received by DMS.
2. DMS locates the fault and prioritises the problem by consulting CIS for the customer data.
3. DMS issues a work order by sending commands to Field Force Management System, which then dispatches the order to a correct work group.
4. Work group reports repairing work status. This is done by Field Force Management System sending a report to DMS. [89]

The described data exchange is implemented with an ESB, and as illustrated in Figure 7.5, all the messages travel through the ESB. The messages are constructed based on the IEC 61968 profiles. [89]

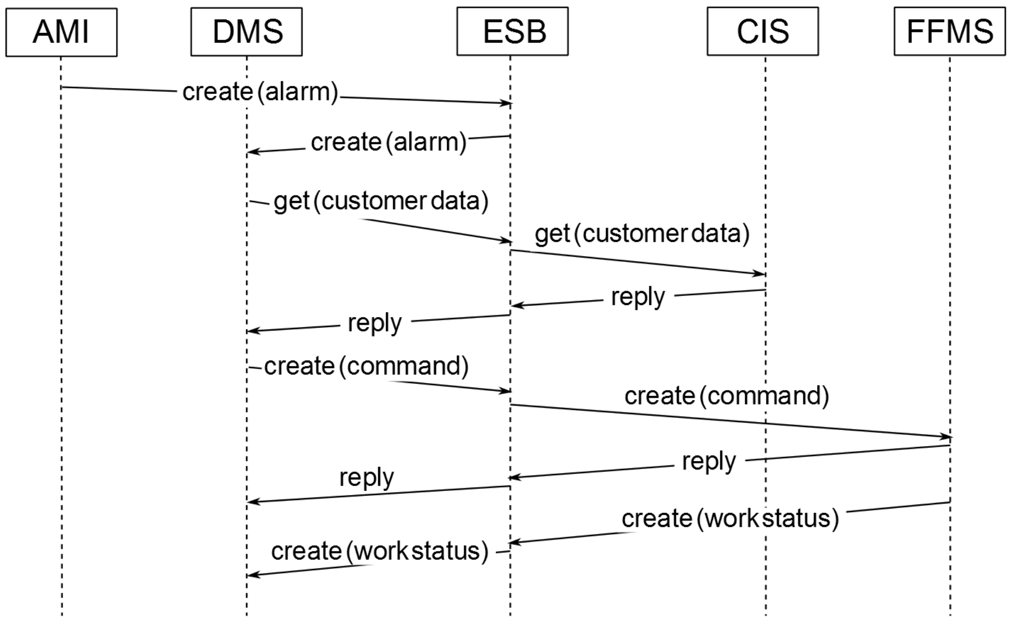


Figure 7.5. *Data exchange in the fault repairing use case, adapted from [89].*

The idea is that when a smart meter alarms about a problem, the DSO systems can automatically locate and prioritise the issue, and assign a field crew for repair work.

7.2.1 Data flows

The data flows in this use case are more complex than in the network model exchange, as Figure 7.6 illustrates.

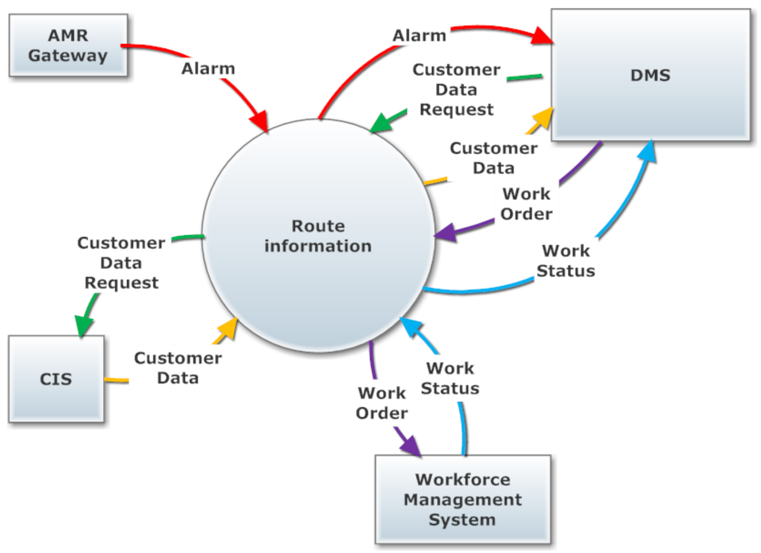


Figure 7.6. *DFD for the fault repairing use case.*

There are some points worth considering in the use case description. The first steps of the use case indicate that some point-to-point way of thinking may still affect how the message exchange is defined. In the description, one point-to-point link is when the AMR Gateway alarms the DMS, another when the DMS consults the CIS for customer information. The CIS then replies with the requested customer information.

However, when using pub-sub messaging, the AMR gateway does not send the alarm directly to the DMS (or any other system). Instead, it publishes it to the hub or bus. The alarm originating from the smart meter probably already has some sort of customer ID, as well as location data, attached to it. The CIS should subscribe to all these alarms coming from AMR gateways. When an alarm is published, the CIS receives it, and then adds customer information based on the customer ID. The CIS then (re)publishes the message (which contains both the alarm and added customer data) to the ESB. This scenario is illustrated in Figure 7.7.

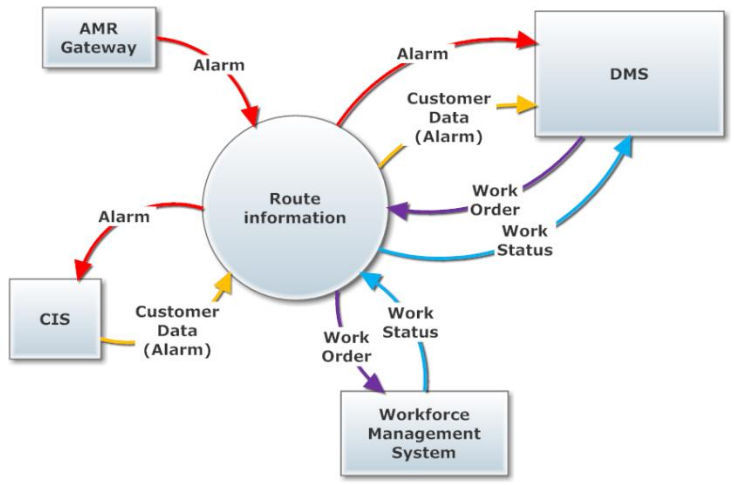


Figure 7.7. *Simplified DFD for the fault repairing use case.*

As the DFD shows, the DMS can, in fact, be subscribed both to the alarms coming directly from the AMR gateway (so it can immediately start to process them) and the "customer data (alarm)" coming from the CIS. This way, there is no need for the DMS to specifically request the customer data. The point is not that requesting the data would be a huge effort. This has more to do with fully embracing the idea of pub-sub messaging. However, building an ESB solution that uses itinerary-based routing will change the nature of the entire messaging. This will be discussed in the lessons learned section.

7.2.2 Information security

This use case brings up more interesting information security issues, as it is dealing with metering data and other things outside the control centre (BizTalk) domain, and also with data from service providers.

Thus, the flows that move across the security/trust boundaries of the DSO, to and from the partner systems, are of specific concern. The flows originating from there, e.g., manipulated metering data can cause disturbances in the system, if widespread misinformation infiltrates the system. Otherwise, the data flow analysis is essentially similar as in the use case A, and detailed analysis for flows will not be discussed here.

7.2.3 Implementation with BizTalk

It is not possible to implement this use case with the available demonstration environment. Apart from the DMS, the endpoint systems are not available for testing.

This case is more complicated than the previous one, with more endpoints and data flows, and analysing it further points out the difference between an EAI type message broker and an ESB solution. Implementing the case as originally described leads to an EAI type of solution: the process is largely controlled by the DMS, with the middleware only handling the messaging. The DMS carries out most of the decision-making and interacting with the user, consulting other systems when necessary. Consulting can be implemented so that the systems provide service interfaces, and the DMS can access the required functionality and information through them. The message flows shown in the DFDs can be implemented as a messaging-only solution in BizTalk, or an orchestration can be used (although not required). Parts of the business logic can be implemented in the orchestration as well. However, in such implementation, the middleware acts as a traditional message broker, and DMS controls the process.

In an ESB solution, the process is initiated by an alarm message that enters the bus (a BizTalk ESBT on-ramp). The appropriate routing steps are first solved, then attached to the message as a routing slip, and then executed. Based on the itinerary, an alarm message first needs to get customer and/or location information. The ESB routes the message (alarm) to a service handling the task (this service is provided by the CIS). Once the required customer data is attached to the message, the alarm needs to be prioritised, so the message is routed to a service handling that task (this service is provided by DMS). The prioritisation service's output is a work order message, which leaves the ESB through an off-ramp and is sent to the WMS.

The status reports sent back from the WMS initiates another messaging process. The WMS could, for example, offer an estimate how long it will take for the crew to repair the fault. Systems that are interested in the updated status will receive the information. This could include, e.g., an SMS notification system that notifies the customer that the fault has been acknowledged and a repair crew is on its way. However, these are options for future research and not discussed here.

7.2.4 Lessons learned

The analysis of this case brought up some of the same issues as the first case. However, this case is more complex than the first one, and additional concerns were identified as well. Creating an essential model should again be the first step. Generally, the idea is that a smart meter raises an alarm, and the DSO information systems can automatically assign a field crew to repair the problem. Once the essential model (the "what") is provided, it can be decomposed in order to show the sub-processes that define in more detail how the use case is carried through. An example is given in Figure 7.8.

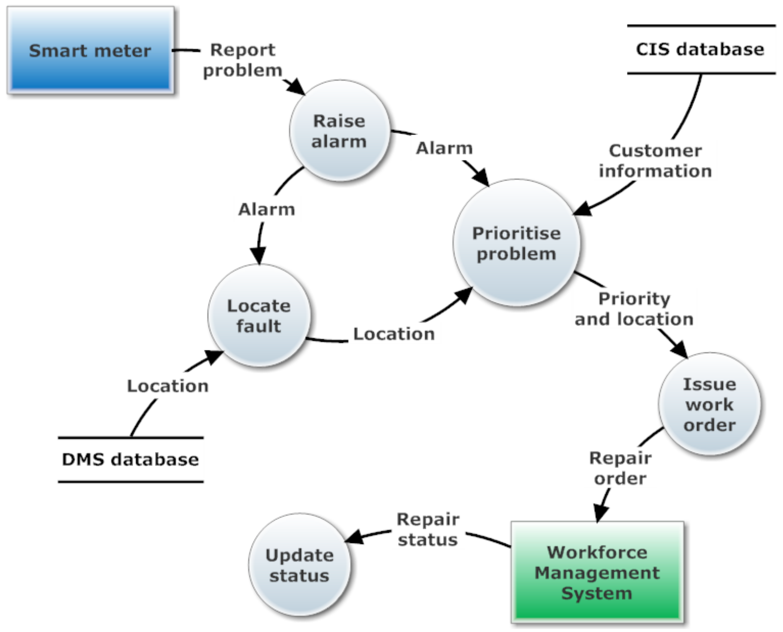


Figure 7.8. *Essential model decomposition in the fault repairing use case.*

This type of detailed analysis defines the use case explicitly, but does not yet specify the technical details. The key design decisions after this step are: 1) which of the data flows are internal to systems, and which are inter-system flows (i.e., implemented by the middleware), and 2) which system will manage the orchestration of the process.

If the orchestration logic resides within one of the existing systems (e.g., the DMS in this use case), the middleware's role is to act as a messaging component. In an ESB solution, much of the logic is implemented in the form of an itinerary. In a message broker scenario, the endpoints can be implemented as services, in alignment with the SOA design principles, but this is not necessary. If an ESB solution with itinerary-based routing is used, the endpoints should have well-defined service interfaces. The services can be built by wrapping the functionalities of the existing systems into services.

7.3 Case C: Active voltage control

The third use case concerns active voltage control. The message flow, described in [89], is as follows:

1. SCADA collects measurement data and status information from the network and large generation units and publishes the data on the ESB.
2. Home Energy Management Systems (HEMSs) measure distributed energy resources (DERs) such as small-scale generation and demand response at customers' homes.
3. Aggregator collects all the measurement data from HEMSs, and then publishes the available DERs for distribution management on ESB.
4. DMS updates the distribution network topology based on status information and the amount of available DERs based on HEMS information gathered from ESB.
5. State Estimator retrieves the measurement data from ESB and executes state estimation. The state estimation results (e.g., maximum and minimum voltages in the network) are forwarded to Coordinated Voltage Controller via ESB.
6. CVC determines whether control actions are needed. In the case of control action for DERs, DMS determines which customers are able to adjust their consumption or generation, and sends control command towards correspondent HEMSs via Aggregator. In the case of control action for the network, CVC calculates new set points for Automatic Voltage Regulator (AVR) of primary transformer tap changer and power factor controller of large generation unit, and sends the new set points to SCADA via ESB. [89]

Figure 7.9 illustrates the systems participating in the use case.

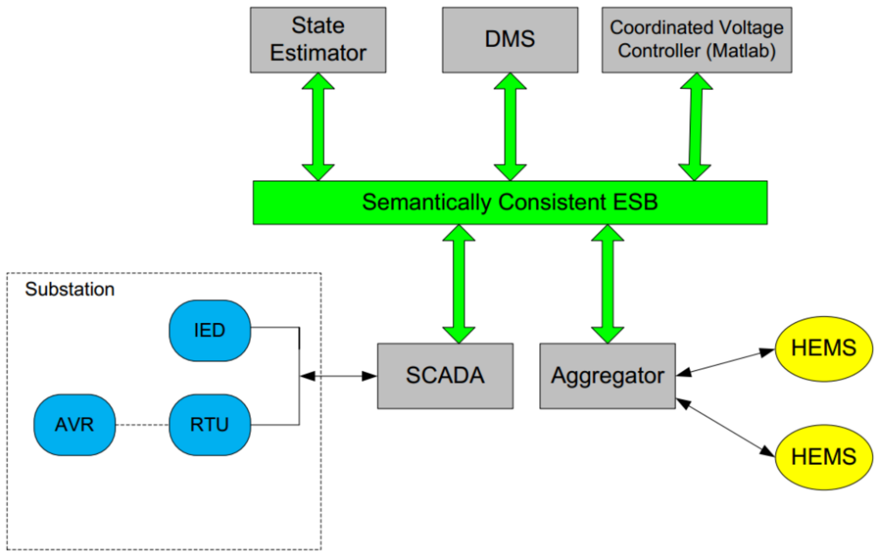

State Estimator
DMS
Coordinated Voltage Controller (Matlab)
Semantically Consistent ESB
Substation
IED
AVR
RTU
SCADA
Aggregator
HEMS
HEMS

Figure 7.9. *Systems participating in the active voltage control use case [89].*

The substation automation components, e.g., the AVRs, Intelligent Electronic Devices (IEDs), and Remote Terminal Units (RTUs), are not directly communicating with the ESB. The SCADA controls the substation components and handles the communication with the ESB. Similarly, the HEMSs only connect to the ESB via an Aggregator.

7.3.1 Data flows

Among the three use cases, this is the most complicated one. It is somewhat different from the two other cases. For example, many of the processes are “housekeeping routines” that run continuously and constantly, instead of having a distinguishable beginning and end. The data flows are illustrated in Figure 7.10.

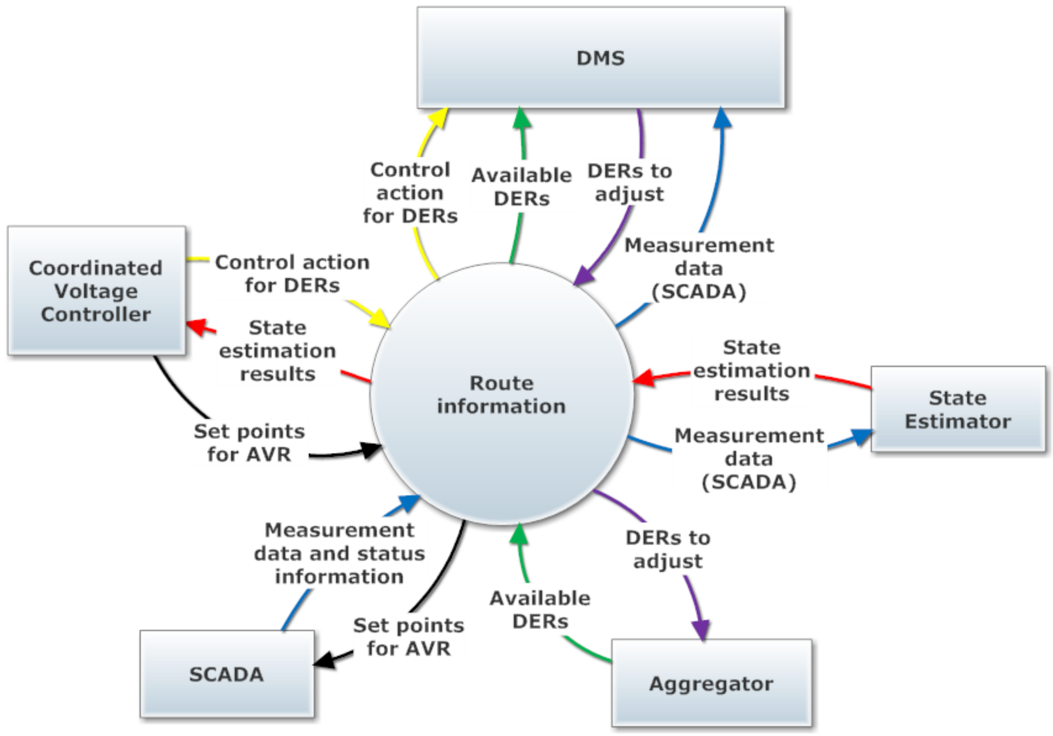


Figure 7.10. *DFD for the active voltage control use case.*

Some systems that are part of the use case are omitted from Figure 7.10, as they are irrelevant from the ESB point-of-view. For example, SCADA and Aggregator need to gather the data from lower-level network components and customer premises. However, the ESB is only aware of the SCADA and Aggregator; the substation components and HEMSs are unknown to it. Within the SGEM project, both the SCADA and DMS are ABB products. These products are quite tightly coupled to each other. They communicate using a direct link rather than through the ESB.

7.3.2 Information security

The flow-based security analysis approach is similarly applicable to this use case as to the previous two cases. What makes this case interesting is that it has more of the aspects of a control system than the other two. The control actions have direct consequences in the physical realm (whether controlling the substation level equipment or DERs at customer premises). This case also has probably the strictest requirements for timeliness.

Depending on how fast the control actions should occur, it may not make sense to implement this case using an ESB. Fluctuations in the voltage may require extremely fast, deterministic, and reliable corrective measures. Enterprise integration platforms are not designed for this sort of communication, and it is unlikely that they can meet the requirements. However, there may additionally be need for a slow-paced (more of trend-based or statistical) control of DERs and substation devices. The real-time requirements for this type of use are likely less demanding, and an ESB could provide sufficient performance. Optionally, two types of middleware solutions could be used: one for lower-level real-time critical systems and use cases, another for higher-level data and control actions.

7.3.3 Implementation with BizTalk

Similar to the use case B, it is not possible to implement this case with the available equipment. Apart from the DMS, the endpoint systems are not available for testing.

Logically, the use case description forms a single sequence. However, the implementation can be divided into sections. The updates coming from SCADA and Aggregator could be their own elements. They are essentially status update messages happening constantly at the background.

The measurement data coming from SCADA could optionally trigger a sequence that can be implemented using an itinerary. The first step in the sequence is a service that performs the state estimation. The results of the state estimation go to a voltage control service, which outputs the required control actions, routed to the DMS or SCADA, depending whether controls are for DERs or AVR. Whether it makes sense to implement this using an ESB depends on the possible hard real-time requirements.

7.3.4 Lessons learned

In this case, a lot of the processes are “housekeeping routines”, performed constantly and automatically in the background. They are constantly on-going, i.e. there is no clear beginning or end point, like in the other cases. This changes the way that the entire process should be coordinated. The implementation is easy if each status update can be treated as an individual message that is simply published into the ESB. If the process requires constantly on-going coordination that links various updates together, the implementation logic is more complicated.

However, the main lesson of this case is the possible introduction of hard real-time requirements. If the processing of a status update, and a resulting control action, require deterministic performance, it is unlikely that an ESB-based, loosely coupled, service-oriented solution is an optimal choice. Using two different buses for communication, based on real-time requirements of the messages may be an option. However, it will make the environment more complex.

8 THE DESIGN AND IMPLEMENTATION OF SECURE INTEGRATION

The guideline presented here aims to combine the lessons learned from the use case analysis, and to point out things to consider in the design of new use cases and improvement of the existing ones. In order to create a more universal approach, the guideline combines the issues that surfaced during this work and generalises some of the main points that were learned.

There are a few important things to keep in mind. First of all, the guideline is very general and primarily offers food for thought. These suggestions do not cover everything or provide any definite answers. The guideline is not implementation-specific; it can be applied to any middleware platform. It is formatted as a list of steps, but the steps are not always separate or discrete. They are intertwined, and it is difficult to categorise a certain task to a specific step on the list. The guideline process is iterative: it is unlikely that the first version of design is optimal. It is also a haphazard approach to analyse just one case and start to implement it. Analysing more than a few cases first is beneficial, as the functionalities that will be needed in many scenarios are probably optimal candidates for services. On the other hand, there is no definite answer as to how many cases are sufficient.

8.1 Create an essential model through business analysis

Defining use cases should begin at a high level of abstraction. This is called business modelling, also known as abstract or essential modelling. The aim of the analysis is to answer the question **what** is done or will be done, omitting the details of **how** it is done. [19] A high-level definition of what is done also answers **why** it is done, in other words, it serves to determine the purpose for the use case.

It is very common for technically oriented people to skip this step and start from a point that is closer to the implementation details (the "how" part). It is a natural tendency to think in terms of technology before explicitly defining the objective and the drivers behind the scenario. In fact, this is unfortunately what happened in this project as well. Both the use case definitions and the integration work started without an explicitly defined essential model. Another reason to exclude this step is that the essential model is often so seemingly simple that it is difficult to grasp its value.

The essential model should focus on the concepts and ignore all the technological details. The same "what" can be achieved with many different technologies, and the "how" is likely to change more frequently than the "what" as technologies develop.

Moreover, the essential model establishes a common ground, an explanation of the purpose of the system, in a language that different stakeholders and technology experts can all understand. Feasible tools for essential modelling are, for example, data flow diagrams and UML use case diagrams. An example of an essential model illustrated with a DFD is given in Figure 8.1. In order to create an accurate essential model, it is crucial to understand exactly what the business objectives of the system are (i.e., what its purpose is, why it is being done). [19]

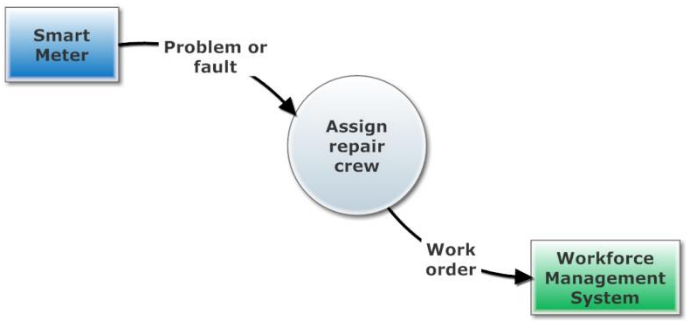


Figure 8.1. *Sample essential model DFD for the fault repairing use case.*

The model could also include a short textual description that defines, on a conceptual level, the state of the system when the use case starts, what triggers the start of the use case, and the desired end result (i.e., the state of the system when the use case is finished). For example, the drop of voltage in the network below a threshold would be the starting point. The use case would be to correct the voltage, and this should happen automatically. The desired end state would be that the voltage is returned to an acceptable value. All this should be expressed using general terms. Technical details are not important yet. This step should also include security requirements (and possible real-time and reliability requirements).

The results of this phase:

1. An essential model describing the purpose of the use case, why it is done. This is given in a common language that is easily understandable for experts in different domains, e.g., a simple DFD representation or UML use case diagram, possibly accompanied by a textual description.
2. General security and priority requirements, i.e. what happens if this use case cannot be performed as intended, or within a specified time limit.

8.2 Define the use case explicitly

The essential modelling describes, in general terms, **what** is done. The next step is to describe the details of **how** things are done. The three use cases used as the starting point of this work signify this level of detail. They include technical details about the system (e.g., identify the systems that participate in the message exchange).

This step can include both a description of how the current system works, and a plan for how the system should work in the future. The scope of coverage depends on the use case. Many of the smart grid functions are entirely new and something that the current system cannot do at all.

This phase should not yet specify the integration architecture or the use of middleware. Instead, focus should be on identifying the required processes and information. It can be achieved, for example, by using data flow diagrams and ignoring how the flows will be implemented. Figure 8.2 illustrates an example DFD.

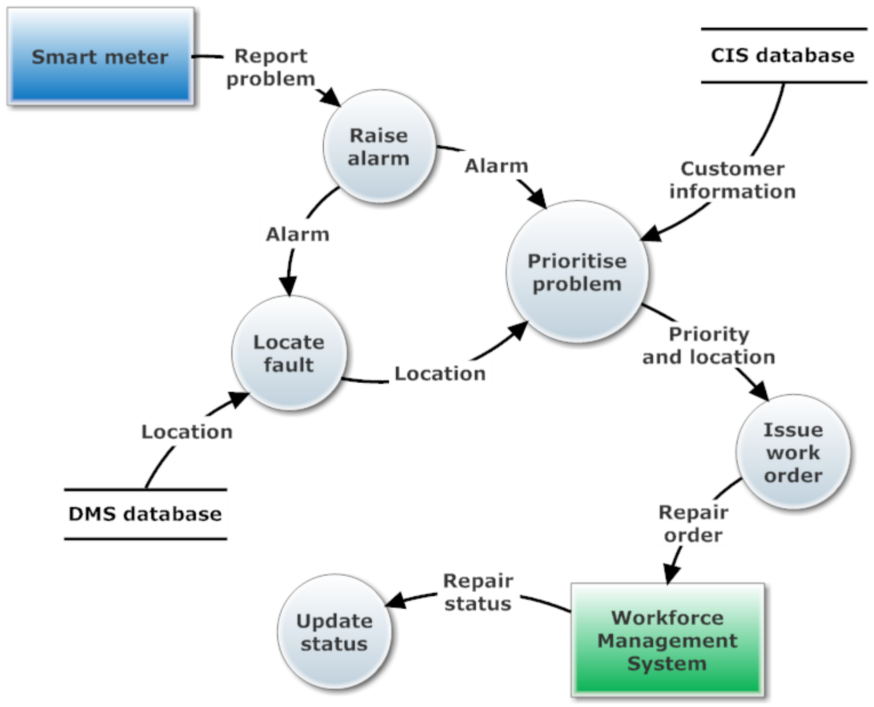


Figure 8.2. *Sub-processes in the fault repairing use case.*

In essence, during this step it is assumed that there is an ideal way for each system to communicate, but its implementation is not yet defined. It is not yet necessary to specify which data flows are internal to the systems and which require inter-system communication (that is, middleware). This supports the creation of an optimal situation, where middleware works as invisible glue between the systems.

This step can greatly benefit from utilising the UML use case modelling principles, rules, and diagrams. Questions that should be answered are, for example, the following: Who or what initiates the use case? (It could be triggered by an event, a user action, or it could be time-based or a continuous housekeeping routine.) What is the degree of automation? Does the process require user intervention? What type of coordination or orchestration is required? (This varies depending on how complex the case is. At this point, it is not necessary to determine which system will take care of the coordination or orchestration; it could be one of the existing systems or the middleware).

The result of this phase: a detailed model that specifies the internal processes in more detail than the essential model, but does not define how the data flows are implemented (i.e., ignores the technical details of message exchange and possible use of middleware).

8.3 Determine the participating systems

This is a simple and very logical step, and tightly intertwined with the previous step. Once the use case is defined in detail, it should be clear which systems will participate. However, it is important to actually list them. This helps to ensure that all the required systems are identified, and that there are no systems mentioned that are not necessary.

The use case definition shows what information and functionality is required. Based on that, it is easy to see which systems contain the necessary information and can perform the required functions. The result of this phase is a list of systems that participate in the use case.

8.4 Define the orchestration of the process

The general use case definition specified **what type of orchestration is needed** to manage the entire transaction. This step will concentrate on **how the orchestration is implemented**. Simple data transfers require very little or no orchestration, as contrasted to more complex, long-running transactions calling for more complicated orchestration.

It is important to define explicitly how much the transaction is orchestrated by the ESB middleware, and what is required from the participating systems (or services). This is a design decision: Should the middleware handle the orchestration, or does the intelligence reside within the endpoint systems? The decision has major consequences to the implementation of the integration. If an endpoint system (e.g., the DMS) handles the orchestration, the integration solution’s role is mainly to function as a message broker. In order to build a service-oriented solution, the endpoint systems should only offer the required functionalities as services and the ESB should handle the orchestration.

Depending on the existing solution, this step may indicate that significant restructuring is required. If the objective is to build an ESB-based SOA solution, restructuring cannot be avoided. For example, if the use case is not new, a lot of the orchestration capabilities may already be implemented in one of the systems. It may be possible to use the existing functionality for orchestration, and use the middleware only for message routing. This way, less change is required, but the result will not be an ESB. The entire orchestration logic should be assigned to the ESB (implemented as an itinerary), or to a separate orchestration service accessed via the ESB.

Possible error conditions and situations should also be considered within this step. How will exceptions be handled and by which system? What if a certain service or system is unavailable? If security will be offered as a service, this is the point for considering its functionality and implementation.

The result of this step is the design decision that determines the role of the middleware. It will be either a simple message broker, or an ESB offering the foundation for an SOA. Based on this decision, the orchestration logic will be explicitly defined and assigned to a specific system that will implement it.

8.5 Define and implement services

Generally, services can either be built starting from scratch, or the functionality of existing systems can be exposed as services in order to let other systems access them. Further, the service "wrapper" can be built into the endpoint system, or it could be implemented by the middleware (e.g., in cases where a legacy system cannot be altered). In a truly service-oriented world, the "participating systems" would instead be "participating services". Any required information or any needed functionality would be built as a well-defined service, accessible through the ESB. The use cases could be implemented by merely combining the required services. This is the fundamental idea of SOA. However, due to the historical weight of monolithic software, the reality is very different in most domains. Most important software products within the utilities industry (e.g., the DMS) probably remain to exist as large, monolithic structures for the foreseeable future. Thus, exposing their existing functionality as services is a good starting point for a move towards SOA.

This step is, again, intertwined with the previous one. The more service-oriented the environment becomes, the better fit an ESB is to handle the overall orchestration. The earlier design decision, along with the appropriate definition of services during this step, largely determines how service-oriented the environment will be. Knowing the guidelines of what makes a good service, it is important to analyse the use case and the participating systems and their functions. The analysis of a variety of use cases will reveal functionality and information that is repeatedly used. These are good candidates for services.

A vital part of this step is to determine whether the service interfaces will offer data and functionality in a CIM format. The goal is that each interface would use CIM. The systems can use legacy formats internally, but the interface should hide this. If the service cannot offer a CIM interface, then the ESB must transform the data between CIM and the provided format, and vice versa.

There are no exact definitions how to specify and construct a service. It is a process that improves along with experience and requires knowledge and understanding of both service orientation and the solution domain (i.e., electric utilities). The process of identifying and defining the services, let alone the various implementation possibilities, will not be discussed in detail here. Most books covering SOA offer details on this topic.

The result of this step: 1) Recognition of information and functionality that should be offered as services. 2) Definition of the services and the interfaces offered. Interfaces should be CIM-based in order to gain full advantage of the integration. 3) Implementation of the services (this is not a trivial task; it requires many design and implementation level decisions, these will not be discussed here).

8.6 Define data flows to be implemented by middleware

The earlier diagram (Figure 8.2) defines many data flows between processes and information storages. Some flows are internal to the endpoint systems, and some are inter-system flows. The latter are the ones that the middleware platform will need to handle, and this step defines how those are implemented.

The earlier design decisions regarding orchestration and services determine largely how this step will be carried out. For an EAI type message broker, as in Figure 8.3, one of the endpoint systems (e.g., the DMS) will orchestrate the process, and the middleware will merely route messages between the systems. However, such a message broker is not necessarily service-oriented, and is definitely not an ESB.

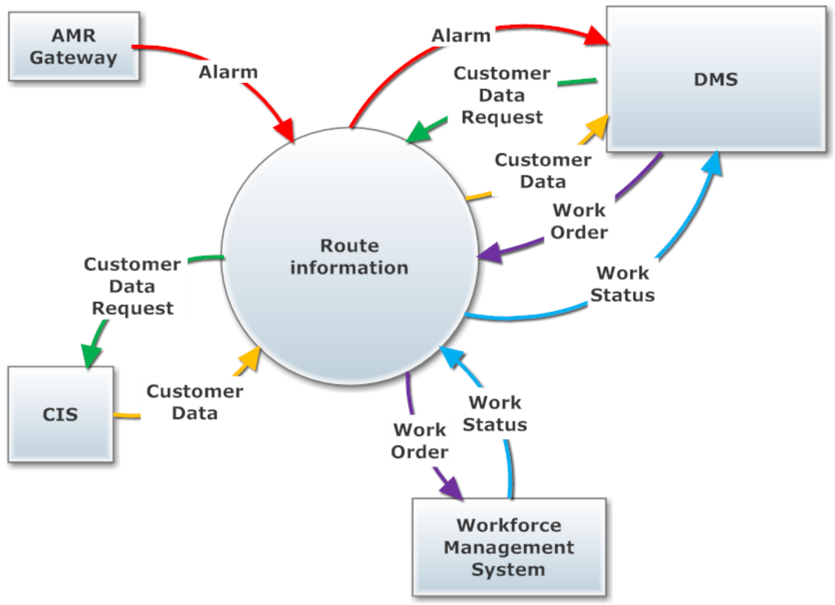


Figure 8.3. *EAI type message broker implementation.*

In a truly service-oriented ESB solution, a message enters the ESB and is then routed through a set of services to produce the desired end result. An ESB type implementation is illustrated in Figure 8.4.

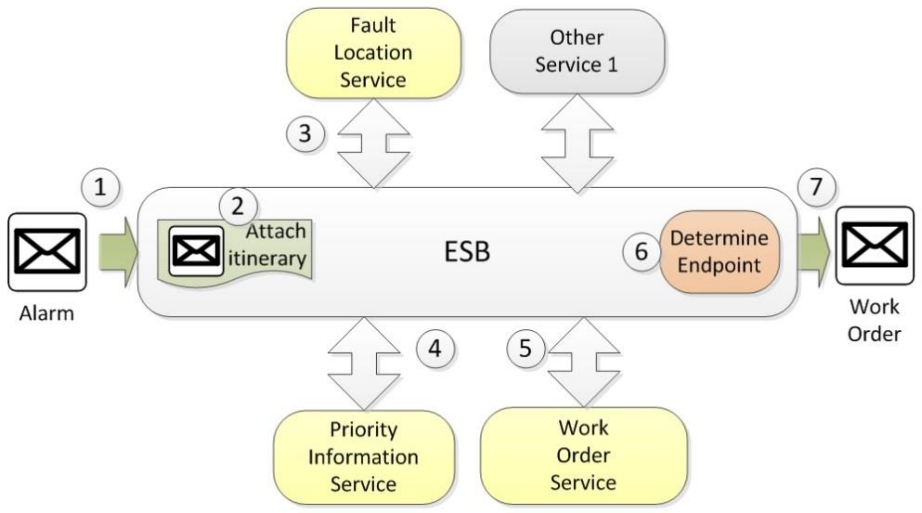


Figure 8.4. *ESB type implementation.*

In the example case, an alarm message enters the ESB, and is routed through the various services necessary to implement the use case. The necessary services provide, for example, fault location, fault prioritisation based on customer information, etc. The end result is a repair order message that exits the ESB and is routed to the WMS (the repair status update message coming back from the WMS will initiate another, similar routing process).

This step clearly shows the way the earlier design decisions regarding orchestration and service-orientation significantly affect how the integration will be implemented. The objective is to build service-oriented environment, and accordingly, this step should define how the ESB routes a message between the appropriate services in order to implement the use case. Regardless of the implementation, a DFD like the one in Figure 8.3 illustrates clearly which data travels through middleware. It is recommended to draw such diagram, as it will be helpful in the next steps.

The result of this step is a concrete definition of which data flows need to be implemented by the middleware and how this will be done. The outcome depends largely on the decisions made earlier.

8.7 Define the information content of data flows

Similar to the third step, this step is simple and logical, yet highly important. Based on the data flows defined in the previous step, it is a straightforward task to define what information each flow contains. Each flow translates into a message that the middleware needs to process.

The content should be in CIM format whenever possible. This is not always easy to implement, but it will greatly increase the interoperability of the entire solution. The result of this step is a listing of the information content of each flow to be handled by the middleware.

8.8 Define information security requirements

The platform security of the participating systems and the middleware itself are extremely important. However, they are outside the scope of this process. Here, the goal is to define information security requirements for the data flows handled by the middleware platform. It is important to observe that the trust boundaries within the environment might change, for example, if a certain service is offered by an external party. Moreover, this process only discusses a small portion of the overall security of the environment: the security of the data while it is moving between systems (i.e., transmitted by the middleware).

Based on their information content, each data flow has specific security requirements (i.e., requirements for confidentiality, integrity, and availability). This step should define those requirements. This is also a logical step to consider the overall quality of service (QoS) requirements for each data flow. Further, it should be analysed whether the combined requirements for the flows are sufficient to meet the general information security requirements for the case as a whole (as defined during the first step, while creating the essential model). Collaboration between a domain expert and an information security professional is strongly recommended for this step. Information security experts have the mind-set and knowledge to ask the right questions that will bring up security requirements. Domain experts understand the system and know how to answer these questions.

The result of this step is the specification of the requirements for the information security of each message. This is achieved by analysing the information content and determining how and why it is valuable.

8.9 Choose information security implementation methods

Once the security requirements are defined, appropriate security implementation methods should be used. This includes both general decisions (e.g., “this data flow needs to be encrypted to provide confidentiality”) and implementation-specific details (e.g., “this middleware offers this type of technologies for encryption”).

If the middleware functions as a conventional message broker, this step is rather straightforward and based on the information content of the data flows. However, the situation may be more complicated for a service-oriented ESB solution. Just as service-orientation itself is a paradigm shift, it implies a paradigm shift in security as well.

The results of this step include the appropriate methods selected to ensure that the information security requirements are met. Based on the requirements defined in the previous step, the selection is first done on a general level, and it can then be further specified, depending on what methods are available on a selected middleware platform.

8.10 Implement the solution

All the necessary information has now been gathered. The final step is to implement the actual orchestration into the selected middleware. Proper technologies available in the middleware platform should be used to ensure that the security requirements are met. The design decisions made earlier will largely determine whether the solution will be more of a traditional EAI message broker, or a truly service-oriented ESB.

A more detailed description of how to apply BizTalk to implement use cases as defined using this guideline would require much more space, and it is not discussed here. However, the BizTalk introduction, together with the demonstration environment that was built, provides a good starting point for the implementation of new use cases using the BizTalk platform.

The result of this step is a functional solution implemented by means of a specific middleware platform.

8.11 Further considerations: testing, maintenance, and modifications

The guideline presented here offers an example process for analysing a use case and serves as a starting point for implementing it on a selected middleware platform. However, it does not offer comprehensive guidance. It is obvious that integrating the various systems in the given environment is a more complex issue than just implementing a few use cases, and there are additional things to consider.

Although the environment is changing at a rather slow pace compared to some other environments, it is not entirely static. The guideline does not take into account how to prepare for these changes. What the correct measures are depends largely on whether the final solution is more of a static EAI message broker, or a truly dynamic ESB solution. However, in both cases, maintenance of the implemented solution should be planned for. Future modifications are inevitable in any environment, and they should be as easy to implement as possible. Another important aspect omitted from the guideline is the testing of the working solution. These further considerations emphasise the fact that the guideline is exactly what the name says: merely guidance.

9 GENERAL RESULTS AND DISCUSSION

The use-case specific results were described as “lessons learned” for each case. The guideline presented in Chapter 8 was a combination of these lessons, provided in a more universally applicable format. This chapter sums up the more general results.

As a concrete result of this work, a demonstration environment now exists. It offers a BizTalk based integration component, which can be used for integrating various DSO ICT systems. The installed environment can be used as a starting point for future use case implementations. The BizTalk introduction provided in this thesis, along with the references, hopefully helps to lower the somewhat steep learning curve of BizTalk.

9.1 Integration and service-orientation

A major aim of this work was to gain understanding of integration architectures (such as EAI and ESB), and their differences. The gathered information is provided as a background theory in this work, but it is useful when considering what type of integration is required for a project. Such basic understanding is crucial and offers a starting point for the comparison of various vendor offerings. It is difficult to choose a proper integration platform without knowledge of what distinguishes an ESB from EAI, for example.

This work does not specify any definite criteria for selecting middleware products. In fact, middleware products are often so complex that it is usually cost-prohibitive to evaluate or test them in detail. However, the basic concepts of, for example, what constitutes an ESB, should be clear when choosing a platform. Otherwise, the comparison cannot really be based on facts, and decisions will be uninformed. Choosing an unsuitable integration platform may cause irreparable damage and make it very difficult to achieve the overall integration goals. Table 9.1 shows the differences between the key properties of integration architectures.

Table 9.1. *Key properties of different integration architectures.*

	Point-to-point	EAI/Hub-and-Spoke	ESB
Scalability	Does not scale well. Useful only in very simple environments with a limited number of nodes.	Scalable design, easier to add more endpoints. Deployment scalable usually only in large increments/ decrements (add/remove hub instances).	Scalable design, easy to add more endpoints. Scalable deployment, can be incremented/decremented gradually. Internally service-oriented.
Routing	Not applicable (end points directly connected).	Static, content or topic based.	Dynamic, itinerary based.
Architecture	Usually ad hoc.	Monolithic	Service-based

The proposed move towards a service-oriented architecture is described as a paradigm shift, and it truly is the case. Fundamental changes are required; SOA is not something that can be “glued on” or attached to the existing architecture. At least, it is unwise to expect major benefits from such an approach. The fact is that the massive software products offered by the largest vendors are inherently monolithic in nature. They represent years (or decades) of development work. The positive aspect is that the products are time-tested with a lot of experience built into them. The negative side is that the massive structures will not turn into flexible, service-oriented products overnight.

This generally holds true for the EAI type message hubs as well. In the integration field, many experts consider these solutions outdated. However, the utilities industry is conservative and slow to change (mainly for good reasons, after all, it is not a called critical infrastructure for nothing). This is probably not the right field to experiment with innovative software architectures. Being considered outdated does not matter if, along with maturity, comes tried and tested reliability. Additionally, many of the EAI platforms are produced by the largest software companies, thus the products are backed up with vast resources, expertise, and product support. The business and product continuity are better guaranteed than in the case of a small company and a novel product. The benefits of an ESB, compared to EAI, also depend a lot on the environment. The DSO ICT environment does change, but it is much more static than, for example, the ICT environment of a retail business, with hundreds or thousands of constantly changing suppliers and business partners. In a rather static environment, the disadvantages of a traditional EAI become less significant, or completely insignificant.

However, this should not be interpreted as a recommendation to ignore the concepts of SOA and ESB. The utilities industry is slow in its movements, but it cannot avoid change. The adoption of the principles of service-orientation and more dynamic software solutions should be constantly developed. Massive products may internally never become fully service-oriented, and it may not be necessary either. However, exposing their key functionality as services, for other systems to use, should be considered. It is worth noting that promoting service-orientation is not necessarily in the interests of the established software vendors, as it would lower the entry barriers and likely increase competition in the field.

An observation that raises some concerns is the possible need for two different middleware platforms. It is quite likely that a higher-level integration solution (that usually aims for maximal throughput) is incapable of matching the most demanding hard real-time requirements. These platforms are not designed or optimised to provide latency that is consistently low (i.e., fast, deterministic responses).

The critical role of a Canonical Data Model has been recognised and well understood for a long time. A CDM is crucial for achieving high levels of interoperability. Without a CDM, the integration will not scale well, and each implementation will always be specific to a certain environment. A CDM will help solve the problem of exponential growth in the number of data transformations between systems.

The CIM appears to be the best solution available for the utilities industry. However, this is only an assumption, based on the fact that there is a growing momentum for developing and applying the CIM and it is backed by major players in the field. Again, legacy systems and large software products will probably not use CIM as their internal data format for a long while, if ever. Yet the development of most important CIM based (service) interfaces should be a key trend. The use of CIM is not a trivial task; the main problem is the size and complexity of the model. It is hard to overcome this issue, because, by definition, the CIM needs to be applicable to numerous tasks.

9.2 Information security

Information security has a crucial role in the Smart Grid. It is vital to take it into account from the start, and build it into the system. As an environment, the Smart Grid is different than any existing system. It combines traditional IT systems, Industrial Control Systems, and home automation, all in an unprecedented scale.

In order to achieve sufficient levels of information security, experts from various fields need to work in close collaboration. Expertise is required in three major fields: the electrical utilities (especially distribution), systems integration, and information security (both in IT and ICS information security).

In general, there are still major challenges in the move towards SOA, but the fundamental changes in the security aspects should be taken into account early on. SOA signifies a paradigm shift in itself, but it also changes the way of approaching information security. Many approaches that work well with traditional applications are ineffective or even counterproductive in a service-oriented environment.

The data flow based security analysis is one approach towards analysing the risks and appropriate security methods. It is particularly suited for a traditional hub-and-spoke integration. It is also useful in terms of an analysis for an ESB solution, but service-orientation and itinerary-based routing requires additional and other types of security analyses.

Performing a comprehensive information security analysis for a few use cases could be beneficial, even if such detailed analyses for all possible cases may be cost-prohibitive. It is obvious that the Smart Grid is fundamentally a domain for the electricity experts. However, co-operation with security experts could help them to increase security awareness and build a security-oriented mind-set. As a result, when designing and implementing new Smart Grid features and usage scenarios, the domain experts would increasingly pay attention to information security aspects as well. Applying even a basic security analysis at an early stage of the process would be highly beneficial; this is promoted by the above guideline.

9.3 BizTalk

The goal of this work was not to qualify or disqualify the use of BizTalk as an integration solution. Properly testing BizTalk (or any middleware platform) against well-defined criteria is beyond what is achievable in one thesis. In this work, most results regarding BizTalk are based on literature, not experiments.

It is impossible to compare BizTalk with other similar products without experience of using those products as well. BizTalk is a complex platform, requiring quite high levels of expertise. With limited experience, it is possible to build BizTalk solutions that work as intended, but they may be far from optimal in terms of performance or other aspects. This is not to say that BizTalk is any more or less complicated than other similar products. In general, BizTalk offers many graphical and relatively easy-to-use tools to assist in otherwise laborious tasks. However, middleware products are complex simply because the field of enterprise integration inherently is complex.

There are two things that may cause problems when considering BizTalk for this type of integration. First, if there are challenging hard real-time requirements, BizTalk may not be the right solution. This is probably true for all IT enterprise integration platforms in general. These platforms are not optimised for that sort of communication. Another possible problem is scalability: compared to a fine-grained service-based deployment of an ESB, BizTalk represents a sort of an “all or nothing” deployment. However, the environment is somewhat static, so this may not be a major problem.

The fact that BizTalk is internally a monolithic hub is the main argument used to disqualify it as an ESB. Further, even though the BizTalk ESB Toolkit offers itinerary-based routing, each step in the itinerary still passes through the MessageBox. In this sense, the hub is still a single point of failure, which could be used as another argument claiming that BizTalk is not an ESB. However, more important than this debate is to understand what is required from the integration solution, and how well a given platform can fulfil the requirements.

Within the scope of the current work, it is not possible to give a detailed, complete guideline for implementing use cases with BizTalk. This may not be practical at all, as the implementation techniques vary so much on depending on the use case. However, the general guideline helps in designing use cases, and the BizTalk chapter provides basic information about the tools that BizTalk offers for implementation. These, and the preinstalled and configured BizTalk environment, will make the task of designing and implementing future use cases easier.

9.4 Discussion

The main goal of this thesis work was to provide concrete examples and guidance on how to integrate DSO ICT systems. Some results were achieved, but much remains to be done. Implementing the use cases was not a straightforward thing to do. As even the environment lacks so many parts, the use cases still require a lot of work. It could also be argued that the implementation of a few use cases would not have proved that the architecture or implementation platform was well designed (i.e., it is a necessary but not sufficient proof). In more pessimistic terms, one might say that the use case analysis and implementation efforts could provide sufficient proof that there are still flaws, unsolved issues, and challenges. And so it did.

Such concepts as service-orientation, publish subscribe messaging, and dynamic routing require certain, sometimes fundamental, changes in thinking. This work offers some ideas how to apply these concepts to the given use cases, which is not always obvious. Thus, this work serves as a sort of a second round of iteration for the use cases (first round being the definitions in [89]). The first iteration represented more the viewpoint of domain experts, that is, how electrical engineers approach the use cases. This analysis hopefully offers a peek into the integration viewpoint, without forgetting the security aspects.

Regarding the guideline for the design and implementation of secure integration, it should be extremely clear that it is what the name implies: no more than guidance. Provision of a definite, comprehensive list of steps, or a very concrete architecture definition, is beyond the scope of a single thesis. This work merely offers some food for thought of what things may be important to consider. The guideline is formatted so as to be understandable for experts from different domains.

The theoretical background part of this work should help to understand some of the integration concepts. The IT world is rapidly and constantly changing, and it is the breeding ground for endless new and innovative concepts, each improving the previous ones (or, at least, marketed to do so). The concepts are often more or less abstract and vague, and vendors usually have somewhat differing definitions for them. This sort of an environment can be extremely confusing.

It is difficult to offer guidance for the selection of a middleware platform. Any organisation in need of integration functionalities is likely to face a choice. Choosing an enterprise integration platform is different from choosing, e.g., an anti-virus software for a home computer. There are not many reviews and test results available that would objectively consider the various options. The products are complex, and evaluating each one would require lots of resources. Thus, a buyer may have to rely on guidance and aid from solution providers and integration consultants. To further add to the problem, vendors may have very different ideas of what some new buzzword means and how their product fits in that description. Given the complexity of the products and the vagueness of the concepts, it is no wonder there are many differing opinions. Finally, the buyer may not thoroughly know their own requirements. For these reasons, it is very challenging to say which integration platform would be the best option for a given case. This work does not even try to do that. However, it does offer basic knowledge of the integration concepts, and this will hopefully help in making informed decisions.

As regards BizTalk, it has been called the most complex software product from Microsoft. That is to say: the most complex software product from the biggest software company on the planet. BizTalk has an installation base of more than 12 000 systems, making it the most widely used middleware platform. Against this background, one thesis probably has very little new to offer. A failed implementation of the test environment should not lead to a decision to reject the use of BizTalk. On the other hand, one more successful case would not offer much additional proof either.

The installation of BizTalk within the scope of this work is not necessarily a realistic one and the workload it handles much less so. Thus, no realistic performance predictions or analysis can be done on the basis of this work. However, as a result of this work, the environment is actually installed and running, and a lot of material has been gathered and some guidance is provided to help continue the work with BizTalk. There are points to consider and limitations regarding the use of BizTalk as an ESB, but there are no obvious reasons to argue that it could not be used.

Throughout the work, information security aspects are emphasised. In addition of the flow-based approach to analysing risks and information security requirements, the theoretical part of the work provides valuable information. The second chapter points out the reasons why the Smart Grid differs from any other environment, and why its information security is extremely challenging. The third chapter concerns the security of SOA, and shows that many of the tried and tested security methods are not applicable in a service-oriented environment. As part of the theoretical background, this information is not new as such. However, the chapters are valuable because the main points are collected in them, and familiarising oneself with them is a good introduction to the topic.

One important result of this work is the emphasis placed on creating proper essential models right in the beginning, in order to provide all participants with the same overall picture. It is very expensive to take experts, for example, in the fields of electricity, information security, and integration, away from their day-to-day work and bring them to the same table. Thus, it is tempting to skip doing this step properly (or at all). It seems that integration results are achieved by getting the integrators engaged and starting their work as soon as possible. However, good planning is nowhere more important than in the world of software engineering. Comprehensive analysis, planning, design and similar steps at the early stages will likely pay off manifold at the end of the project. There is nothing new in saying this, but these steps are still easily neglected.

Essential modelling helps to ensure that all aspects are taken into account from the ground up. The temptation to get started and achieve concrete results is great. However, creating software is different from creating something physical or tangible, and the concrete results can fall off as it becomes clear that the wrong thing was being built. Software projects are notorious of going overtime and over-budget, or even failing entirely. As impossible as it may seem, the situation may be even worse with the middleware or integration projects. The guideline for the design and implementation of secure integration, as presented in this work, does not claim to offer new or exhaustive solutions, but hopefully it will contribute to future efforts in building secure Smart Grid integration.

10 CONCLUSION

During the course of this work, it became clear that the electrical utilities industry is conservative and changes slowly. This is understandable, as the reliability of the electric grid is critically important. Customers have grown accustomed to a constant supply of electricity, and the modern society relies on it. However, new models of producing and consuming electricity are required, and this calls for a smarter grid.

Efficient and secure integration of information systems, especially in the distribution domain, is vital for the Smart Grid. A service-oriented architecture that facilitates an ESB as the communication backbone is a potential option for such integration. However, ESB or SOA are no silver bullets. Neither are they something that can be attached or added to the existing environment, so as to solve all the integration issues. They are concepts that call for a paradigm shift. In order to be implemented successfully, they require detailed analysis and design. Further, they will change some of the assumptions that many of the commonly used information security methods rely on. Consequently, a change in the information security thinking is required as well.

The development of the Common Information Model is critically important in enabling scalable integration in the Smart Grid. The work is well underway, but it is not trivial to use the CIM in practice. The main challenge, which is hard to overcome, is that the model is so complex.

As part of this work, a demonstration environment was built, in order to test these concepts in practice. The BizTalk integration component now exists, but many other parts of the environment still need to be implemented. The limitations of the environment made it difficult to implement the test use cases. However, the analysis indicated that there are other fundamental issues that should be clarified before proceeding with implementation.

Based on the lessons learned from analysing the use cases, a general guideline was created. It does not aim to be a comprehensive set of rules, and the aspects presented are not necessarily new as such. However, the guideline should make clear what a secure integration based on modern integration paradigms means and requires. It points out what needs to change within the current environment in order to move towards service-orientation and ESB based integration. Moreover, it emphasises the importance of information security in the Smart Grid.

The installed platform, along with the guideline, should make the task of improving the demonstration environment and implementing more use cases easier. As a general recommendation, such work should include experts from various fields and requires their close collaboration.

REFERENCES

[1] Enterprise Service Bus (ESB) and SAP. [WWW]. 16.5.2008. [Referred: 19.2.2013]. Available at: http://rsol08.wordpress.com/2008/05/16/esb-and-sap/.

[2] ABB. MicroSCADA Pro DMS 600 4.4 Operation Manual. [PDF]. A/2012. 24.2.2012. [Retrieved: 14.6.2012]. Available: http://www05.abb.com/global/scot/scot229.nsf/veritydisplay/590b46eca9a49e15c12579c9004d8b6b/$file/DMS600_Operation%20Manual_757319_ENa.pdf.

[3] ABB. MicroSCADA Pro DMS 600 4.4 System Overview. [PDF]. A/2012. 24.2.2012. [Retrieved: 14.6.2012]. Available: http://www05.abb.com/global/scot/scot229.nsf/veritydisplay/4ad10dda9d19c6cdc12579c9004d551c/$file/DMS600_System%20Overview_757322_ENa.pdf.

[4] ABB. MicroSCADA Pro for Network Control and Distribution Management. [WWW]. [Referred: 23.10.2012]. Available at: http://www.abb.com/product/db0003db004281/c12573990068e57cc1256ebc005454be.aspx.

[5] ABB. MicroSCADA Pro for Network Control and Distribution Management. [PDF]. 2010. [Retrieved: 14.6.2012]. Available: http://www05.abb.com/global/scot/scot387.nsf/veritydisplay/95ac6dd9e5569804c1257aef005539be/$file/1MRS756253_C_en_MicroSCADA_Pro_for_network_control_and_distribution_management.pdf.

[6] Ackermann T., Andersson G. & Söder L. Distributed generation: a definition. Electric Power Systems Research 57(2001)3, pp. 195-204.

[7] Adleman, H. & Haigh, D.C. Dynamic integration foundation for the utilities industry. PICA '99. Proceedings of the 21st 1999 IEEE International Conference on Power Industry Computer Applications, 1999. pp. 109-114.

[8] Ahonen, P. TITAN-käsikirja. VTT:n päätuloksia Tekesin Turvallisuusohjelman TITAN-projektissa. Espoo 2010, VTT, VTT Tiedotteita - Research Notes 2545. 152 s.

[9] Arcitura Education. The Service-Orientation Design Paradigm. [WWW]. [Referred: 20.2.2013]. Available at: http://serviceorientation.com/index.php/serviceorientation/p3.

[10] Arcitura Education. Service-orientation Principles: Services. [WWW]. [Referred: 20.2.2013]. Available at: http://serviceorientation.com/index.php/serviceorientation/p1.

[11] Bacon, J., Eyers, D., Moody, K. & Pesonen, L. Securing Publish/Subscribe for Multi-domain Systems. Middleware 2005: ACM/IFIP/USENIX 6th International Middleware Conference, Grenoble, France, November 28 - December 2, 2005. Proceedings. 2005, Springer Berlin Heidelberg. pp. 1-20.

[12] Bakker, L. Goodbye Hub-and-Spoke, Hello ESB? Integration Architecture With BizTalk 2004. [WWW]. .NET Developer's Journal, SYS-CON Media. 12.9.2005. [Referred: 26.8.2012]. Available at: http://dotnet.sys-con.com/ node/121831.

[13] Beckner, M. BizTalk 2010 Recipes: A Problem-Solution Approach. USA 2010, Apress. 592 p.

[14] Bennett S. Insecurity in the Supply of Electrical Energy: An Emerging Threat? The Electricity Journal 24(2011)10, pp. 51-69.

[15] Birman, K.P., Ganesh, L. & van Renesse, R. Running Smart Grid Control Software on Cloud Computing Architectures. Workshop on Computational Needs for the Next Generation Electric Grid, Ithaca, NY, April 19-20, 2011. Ithaca, NY 2011, Department of Computer Science, Cornell University. 28 p.

[16] Bishop, M. Computer Security - Art and Science. USA 2003, Pearson Education. 1084 p.

[17] Boin A. & McConnell A. Preparing for Critical Infrastructure Breakdowns: The Limits of Crisis Management and the Need for Resilience. Journal of Contingencies and Crisis Management 15(2007)1, pp. 50-59.

[18] Brown, K. BizTalk Server 2010 R2 - What's New? 26.9.2012, Microsoft. Presentation at BizTalk User Group Finland event.

[19] Brown, D. The How To of Essential Modelling. [PDF]. Melbourne, Australia, IRM Training Pty Ltd. 28.5.2008. [Retrieved: 23.10.2012]. Available: http://www.irm.com.au/papers/How_To_of_Essential_Modelling.pdf;.

[20] Brown, P.C. Implementing SOA: Total Architecture in Practice. Stoughton, MA, USA 2008, Pearson Education. 699 p.

[21] Brown, P.C. Succeeding with SOA: Realizing Business Value Through Total Architecture. Stoughton, MA, USA 2007, Pearson Education. 244 p.

[22] Cepa, L., Kocur, Z. & Vodrazka, J. The methodology for the selection of ICT technologies for Smart Grids. Proceedings of the 35th International Conference

on Telecommunications and Signal Processing (TSP), Prague, 3.-4.7.2012. 2012, pp. 1-5.

[23] Chappell, D. Enterprise Service Bus. Sebastopol, California 2004, O'Reilly. 247 p.

[24] Chappell, D. ESB Myth Busters: 10 Enterprise Service Bus Myths Debunked. [WWW]. SOA World Magazine, SYS-CON Media. 25.5.2005. [Referred: 27.6.2012]. Available at: http://soa.sys-con.com/node/48035.

[25] Chetty, J. & Coetzee, M. Towards an information security framework for service-oriented architecture. Information Security for South Africa (ISSA), 2010, 2010, pp. 1-8.

[26] Chisholm, M. Six Architectural Styles of Data Hubs. [WWW]. TechTarget. 6.8.2008. [Referred: 19.2.2013]. Available at: http://www.b-eye-network.com/view/8109.

[27] Cluster for Energy and Environment, CLEEN. CLEEN - Cluster for Energy and Environment. [WWW]. [Referred: 12.6.2012]. Available at: http://www.cleen.fi/en/.

[28] Cluster for Energy and Environment, CLEEN. List of SGEM program public deliverables. [WWW]. [Referred: 11.2.2013]. Available at: http://www.cleen.fi/en/sgem/public_deliverables.

[29] Cluster for Energy and Environment, CLEEN. SGEM - Smart Grids and Energy Markets. [WWW]. [Referred: 12.6.2012]. Available at: http://www.cleen.fi/en/sgem.

[30] Conklin W.A. Control systems personnel are from Mars; IT personnel are from Venus. International Journal of Critical Infrastructure Protection 4(2011)2, pp. 76-77.

[31] Cybersoft. Cybersoft Network Manager. [WWW]. 2011. [Referred: 16.2.2013]. Available at: http://www.cybersoft.fi/tuotteet.html.

[32] Cárdenas, A.A. & Safavi-Naini, R. Security and Privacy in the Smart Grid. In: Das, S.K., Kant, K. & Zhang, N. (editors). Handbook on Securing Cyber-Physical Critical Infrastructure. USA 2012, Morgan Kaufmann/Elsevier. pp. 637-654.

[33] Davis, D. & Parashar, M.P. Latency Performance of SOAP Implementations. Proceedings of 2nd IEEE/ACM International Symposium on Cluster Computing

and the Grid, 2002, Berlin, Germany, 21-24.5.2002. 2002, Institute of Electrical and Electronics Engineers. pp. 407-414.

[34] Devi, Y. Exploring Synergy between SOA Cloud Computing. [WWW]. [Referred: 25.10.2012]. Available at: http://www.siliconindia.com/guestcontributor/guestarticle/175/Exploring_Synergy_between_SOA_Cloud_ Computing_Yogesh _Devi.html.

[35] Douglass, B.P. Real Time UML: Advances in the UML for Real-Time Systems. 3rd edition. Boston, MA, USA 2004, Addison-Wesley. 694 p.

[36] Drewry, T. Data Flow Diagrams. [WWW]. October 2005. [Referred: 24.10.2012]. Available at: http://www.cems.uwe.ac.uk/tdrewry/dfds.htm.

[37] Dunphy, G., Campos, H., Kaufman, S., Kelcey, P., Moukhnitski, S. & Peterson, D.H. Pro BizTalk 2009. USA 2009, Apress. 740 p.

[38] Ekanayake, J., Liyanage, K., Wu, J., Yokoyama, A. & Jenkins, N. Smart Grid: technology and applications. 2012, Wiley. 277 p.

[39] Electric Power Research Institute, EPRI. The Common Information Model for Distribution: An Introduction to the CIM for Integrating Distribution Applications and Systems. Palo Alto, CA 2008, EPRI, Technical Update report 1016058. 98 p.

[40] Electric Power Research Institute, EPRI. Common Information Model Primer. Palo Alto, CA, USA 2011, EPRI, 1024449. 148 p.

[41] Electric Power Research Institute, EPRI. Enterprise Service Bus Implementation Profile. Palo Alto, CA, USA 2009, EPRI, Integration Using IEC 61968 1018795. 88 p.

[42] Elovaara, J. & Haarla, L. Sähköverkot II - Verkon suunnittelu, järjestelmät ja laitteet. Helsinki 2011, Gaudeamus Helsinki University Press. 551 s.

[43] Energiateollisuus. Älykäs verkko eli Smart Grid. [WWW]. [Referred: 11.2.2013]. Available at: http://energia.fi/sahkomarkkinat/sahkoverkko/alykas-verkko.

[44] Erl, T. Exclusive SOA Web Services Journal Briefing - Thomas Erl on SOA. [WWW]. SYS-CON Media. 29.10.2005. [Referred: 20.2.2013]. Available at: http://thomaserl.sys-con.com/node/136190.

[45] Eugster P.T., Felber P.A., Guerraoui R. & Kermarrec A. The many faces of publish/subscribe. ACM Computing Surveys 35(2003)2, pp. 114-131.

[46] European Electricity Grid Initiative, EEGI. The European Electricity Grid Initiative Roadmap and Implementation Plan. [PDF]. version V2. 25.5.2010. Available: http://www.smartgrids.eu/documents/EEGI/EEGI_Implementation _plan _May%202010.pdf.

[47] European Industry Association, Information Systems Communication Technologies Consumer Electronics, EICTA. EICTA Interoperability White Paper. 2004.

[48] European Network and Information Security Agency, ENISA. Appropriate security measures for smart grids: Guidelines to assess the sophistication of security measures implementation. 2012, 84 p.

[49] Fiege, L., Zeidler, A., Buchmann, A., Kilian-Kehr, R., Mühl, G. & Darmstadt, T. Security Aspects in Publish/Subscribe Systems. Third Intl. Workshop on Distributed Event-based Systems (DEBS'04), 2004, Institute of Electrical and Electronics Engineers.

[50] Flanders, J. BizTalk ESB Toolkit Core Components and Examples. [DOC]. Microsoft. 26.2.2010. [Retrieved: 7.8.2012]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=14293.

[51] Flick, T. & Morehouse, J. Securing the Smart Grid: Next Generation Power Grid Security. Burlington, MA, USA 2011, Elsevier. 290 p.

[52] Forsström, S. Sähköverkkoyhtiön tietojärjestelmien kehittäminen. Diplomityö. Espoo 2007. Teknillinen Korkeakoulu. 129 s.

[53] GridWise Architecture Council, GWAC. GridWise Interoperability Context-Setting Framework. [PDF]. v.1.1. March 2008. [Retrieved: 17.12.2012]. Available: http://www.gridwiseac.org/pdfs/interopframework_v1_1.pdf.

[54] GridWise Architecture Council, GWAC. Publications. [WWW]. [Referred: 17.12.2012]. Available at: http://www.gridwiseac.org/about/publications.aspx.

[55] Haikala, I. & Märijärvi, J. Ohjelmistotuotanto. 11. painos. Jyväskylä 2006, Talentum. 440 s.

[56] Hedberg, J., Weare, K. & la Cour, M. Microsoft BizTalk Server 2010 (70-595) Certification Guide. 2012, Packt Publishing. 500 p.

[57] Hohpe, G., Woolf, B. Canonical Data Model. [WWW]. 2003. [Referred: 23.8.2012]. Available at: http://www.enterpriseintegrationpatterns.com/CanonicalDataModel.html.

[58] Hohpe, G. Enterprise Integration Patterns. [WWW]. 2012. [Referred: 24.9.2012]. Available at: http://www.enterpriseintegrationpatterns.com/index.html.

[59] Hohpe, G. Hub and Spoke [or] Zen and the Art of Message Broker Maintenance. [WWW]. 12.10.2003. [Referred: 24.8.2012]. Available at: http://www.enterprise integrationpatterns.com/ramblings/03_hubandspoke.html.

[60] Hohpe, G., Woolf, B. Message Translator. [WWW]. 2003. [Referred: 19.7.2012]. Available at: http://www.eaipatterns.com/MessageTranslator.html.

[61] Hohpe, G., Woolf, B. Pipes and Filters. [WWW]. 2003. [Referred: 9.1.2013]. Available at: http://www.eaipatterns.com/PipesAndFilters.html.

[62] Hohpe, G., Woolf, B. Publish-Subscribe Channel. [WWW]. 2003. [Referred: 22.2.2013]. Available at: http://www.eaipatterns.com/PublishSubscribe Channel.html.

[63] Hohpe, G., Woolf, B. Routing Slip. [WWW]. 2003. [Referred: 2.10.2012]. Available at: http://www.enterpriseintegrationpatterns.com/RoutingTable.html.

[64] Institute of Electrical and Electronics Engineers, IEEE. IEEE Guide for Smart Grid Interoperability of Energy Technology and Information Technology Operation with the Electric Power System (EPS), End-Use Applications, and Loads. [PDF]. IEEE Standards Coordinating Committee 21. 10.9.2011.

[65] InterPSS. InterPSS OpenCIM Datasheet. [PDF]. [Retrieved: 18.1.2013]. Available: http://commondatastorage.googleapis.com/opencim/download/opencim_datasheet.pdf.

[66] InterPSS. InterPSS OpenCIM User's Guide. [PDF]. March 2010. [Retrieved: 5.12.2012]. Available: http://commondatastorage.googleapis.com/opencim/download/InterPSS_OpenCIM_UsersGuide_v1.2.1.pdf.

[67] InterPSS. OpenCIM - details. [WWW]. [Referred: 5.12.2012]. Available at: https://sites.google.com/a/interpss.com/opencim/Home/interpss-opencim.

[68] InterPSS. OpenCIM - overview. [WWW]. [Referred: 5.12.2012]. Available at: https://sites.google.com/a/interpss.com/opencim/Home.

[69] International Electrotechnical Committee, IEC. IEC 61968-1 Application integration at electric utilities - System interfaces for distribution management - Part 1: Interface architecture and general requirements. [PDF]. IEC 61968-1:2003(E). 2003.

[70] International Electrotechnical Committee, IEC. IEC Smart Grid Standardization Roadmap. [PDF]. version 1.0. IEC Standardization Management Board Smart Grid Strategic Group (SG3). June 2010. Available: http://www.iec.ch/smartgrid/downloads/sg3_roadmap.pdf.

[71] International Electrotechnical Committee, IEC. Part 1: Reference Architecture for Power System Information Exchange. 2011, IEC 62357 TC57 Architecture 62357-1. 117 p.

[72] Jin, S. SOA and Cloud Computing: Are They The Same? [WWW]. VMware vCloud Blog. 21.4.2010. [Referred: 22.10.2012]. Available at: http://blogs.vmware.com/vcloud/2010/04/soa-and-cloud-computing-are-they-the-same.html.

[73] Johnston, C. Smart Grid and Cloud Computing: The Vast Enterprise Vision for America's Aging Utility Grid. [WWW]. AT&T. 21.6.2011. [Referred: 22.10.2012]. Available at: http://networkingexchangeblog.att.com/enterprise-business/smart-grid-and-cloud-computing-the-vast-enterprise-vision-for-ameri
cas-aging-utility-grid/.

[74] Järventausta, P. Introduction to Smart Grids. Tampere 2012, Tampere University of Technology. Lecture slides.

[75] Järventausta P., Repo S., Rautiainen A. & Partanen J. Smart grid power system control in distributed generation environment. Annual Reviews in Control 34(2010)2, pp. 277-286.

[76] Kanneganti, R. & Chodavarapu, P. SOA Security. USA 2008, Manning Publications. 483 p.

[77] Keski-Keturi, R. Implementing the IEC Common Information Model for Distribution System Operators. Master's thesis. Tampere 2011. Tampere University of Technology. 91 p.

[78] Knapp, E.D. Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems. USA 2011, Syngress. 341 p.

[79] Koivisto, H., Seppälä, J. & Salmenperä, M. Network-based Automation. Tampere 2012, Tampere University of Technology. Lecture slides.

[80] Korteila M. Supra sukeltaa viimein sähköverkkoon. TIEDE 31(2011)4, ss. 32-35.

[81] Koto, A. Tietojärjestelmien väliset rajapinnat sähkönjakeluverkon käyttötoiminnassa. Diplomityö. Tampere 2010. Tampereen Teknillinen Yliopisto. 104 s.

[82] Kumagai J. A Battery as Big as the Grid. IEEE Spectrum (intl.) 49(2012)1, pp. 39-40.

[83] Lakervi, E. & Partanen, J. Sähkönjakelutekniikka. 3. painos. Helsinki 2009, Gaudeamus Helsinki University Press. 295 s.

[84] Levy, D. Smart Grid: Cloud-Oriented Architecture in Action. [WWW]. 3.7.2012. [Referred: 29.10.2012]. Available at: http://www.zapthink.com/2012/07/03/smart-grid-cloud-oriented-architecture-in-action/.

[85] Linthicum, D. S. SOA cloud computing relationship leaves some folks in a fog. [WWW]. GCN. 9.3.2009. [Referred: 19.2.2013]. Available at: http://gcn.com/articles/2009/03/09/guest-commentary-soa-cloud.aspx.

[86] Liu, Y., Gorton, I. & Zhu, L. Performance Prediction of Service-Oriented Applications based on an Enterprise Service Bus. COMPSAC 2007. Proceedings of 31st Annual International Computer Software and Applications Conference, 2007. pp. 327-334.

[87] Loesgen, B., Young, C.W., Eliasen, J., Colestock, S., Kumar, A. & Flanders, J. BizTalk Server 2010 Unleashed. Indiana, USA 2011, Sams. 840 p.

[88] Lu, S. & Repo, S. D6.10.1: Demonstration of using load flow and fault current calculation software tool with CIM interface and NIS database - Survey. Third version. Tampere University of Technology. 2.5.2012.

[89] Lu, S. & Repo, S. Examples of inter-application communications in a DSO. 8 p.

[90] McMorran, A.W., Lincoln, R.W., Taylor, G.A. & Stewart, E.M. Addressing misconceptions about the Common Information Model (CIM). Power and Energy Society General Meeting, 2011 IEEE, 2011, pp. 1-4.

[91] McMorran, A.W. An Introduction to IEC 61970-301 & 61968-11: The Common Information Model. [PDF]. Glasgow, UK, University of Strathclyde. January, 2007. [Retrieved: 9.8.2012]. Available: https://cimphony.com/cim-intro.pdf.

[92] Microsoft. BizTalk Server 2010 Database Infrastructure Poster. [PDF]. [Retrieved: 8.8.2012]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=443.

[93] Microsoft. Installing BizTalk Server 2010 and BAM in a Multi-Computer Environment. [DOC]. July 2011. [Retrieved: 27.11.2012]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=11503.

[94] Microsoft. Installing BizTalk Server 2010 on Windows Server 2008 R2 and 2008. [DOC]. October 2011. [Retrieved: 28.6.2012]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=11503.

[95] Microsoft. Microsoft BizTalk ESB Toolkit 2.1. [CHM]. 12.6.2010. [Retrieved: 29.11.2012]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=11847.

[96] Microsoft. Microsoft BizTalk Server 2010 Help. [CHM]. 10.6.2011. [Retrieved: 25.6.2012]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=11503.

[97] Microsoft. Microsoft BizTalk Server 2010 Technical Overview. [DOC]. 9.10.2010. [Retrieved: 25.8.2012]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=24535.

[98] Microsoft. Troubleshooting BizTalk Server 2010 Setup. [DOC]. September 2010. [Retrieved: 28.6.2012]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=11503.

[99] Morgenthal, J. Cloud Computing Realigns Role of Service Oriented Architecture. [WWW]. 27.7.2011. [Referred: 25.10.2012]. Available at: http://www.infoq.com/articles/Cloud-Realigns-SOA.

[100] National Energy Technology Laboratory. A Systems View of the Modern Grid. [PDF]. version 2.0. U.S. Department of Energy, Office of Electricity Delivery and Energy Reliability. January 2007. [Retrieved: 2.11.2012]. Available: http://www.netl.doe.gov/smartgrid/referenceshelf/whitepapers/ASystemsView oftheModernGrid_Final_v2_0.pdf.

[101] National Institute of Standards and Technology, NIST. NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 1.0. 2010, National Institute of Standards and Technology, NIST Special Publication 1108. 145 p.

[102] National Institute of Standards and Technology, NIST. NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 2.0. 2012, National Institute of Standards and Technology, NIST Special Publication 1108R2.

[103] National Rural Electric Cooperative Association, NRECA. MultiSpeak. [WWW]. [Referred: 19.7.2012]. Available at: http://www.multispeak.org/ Pages/default.aspx.

[104] National Rural Electric Cooperative Association, NRECA. MultiSpeak and the NIST Smart Grid. [WWW]. [Referred: 19.7.2012]. Available at: http://www.m ultispeak.org/about/SmartGrid/Pages/MultiSpeakandtheNISTSmartGrid.aspx.

[105] Nikunen, J. Security Considerations on Wide Area Networking Industrial Solutions. Master's thesis. Tampere 2001. Tampere University of Technology. Department of Automation, 52 p.

[106] Nokia Siemens Networks. Common interests, common platform. Open EMS Suite gets the industry talking. [PDF]. 10/2007. [Retrieved: 14.8.2012]. Available: http://www.nokiasiemensnetworks.com/sites/default/files/ document/Open_EMS_Suite_Brochure.pdf.

[107] Nokia Siemens Networks. Functional Overviews - Open EMS Suite: a multi-technology EMS platform. [WWW]. [Referred: 10.2.2013]. Available at: http://www.nokiasiemensnetworks.com/ems-platform-multi-technology.

[108] Närman, P., Gammelgård, M. & Nordström, L. A Functional Reference Model for Asset Management Applications Based on IEC 61968-1. Proceedings of Nordic Distribution and Asset Management Conference, Stockholm, Sweden, August 2006. 2006, 12 p.

[109] O'Brien, R. Integration Architecture Explained. [WWW]. [Referred: 19.2.2013]. Available at: http://russellobrien.hubpages.com/hub/Integration-Architecture-Explained.

[110] Paananen, K. Information security in Smart Grid demonstration environment. Master's thesis. Tampere 2012. Tampere University of Technology. 97 p.

[111] Papazoglou, M.P. Web Services: Principles and Technology. UK 2008, Pearson Education. 752 p.

[112] Pfleeger, C.P. & Pfleeger, S.L. Security in computing. 4th edition. Westford, MA, USA 2007, Pearson Education. 845 p.

[113] Pitkänen, M., Haavisto, J. & Ollila, M. Tukijärjestelmäliityntä - Tuki- ja käytönvalvontajärjestelmien välisen tiedonsiirron yhteiskäytäntö. Tampere 1997, Tampereen teknillinen korkeakoulu, Sähkövoimatekniikka 7-97. 33 s.

[114] Pitkänen, M., Kärenlampi, M., Verho, P., Järventausta, P. & Partanen, J. Valvomon tietojärjestelmien välisen tiedonsiirron kehittäminen. Tampere 1997, Tampereen teknillinen korkeakoulu, Sähkövoimatekniikka 6-97. 56 s.

[115] Pohjonen, R. Tietojärjestelmien kehittäminen. 2. painos. Jyväskylä 2002, Docendo Finland Oy. 178 s.

[116] Prince, B. Industrial Control Systems are 10 Years Behind Enterprise IT in Security, Say Experts. [WWW]. Securityweek. 23.11.2011. [Referred: 18.2.2013]. Available at: http://www.securityweek.com/industrial-control-systems-are-10-years-behind-enterprise-it-security-say-experts.

[117] Rapo R. Kestääkö verkko sähköautojen rasituksen? Prosessori 33(2011)1-2, s. 18.

[118] Repo, S., Rautiainen, A., Koto, A., Lu, S. & Valavaara, T. Demonstration environment for smart grid applications. [PDF]. First version. Tampere University of Technology, Department of Electrical Energy Engineering. 30.9.2010. [Retrieved: 15.6.2012]. Available: http://webhotel2.tut.fi/units/set/research/inca-public/tiedostot/Raportit/OES_ThereGate_demo.pdf.

[119] Repo, S., Rautiainen, A., Koto, A., Lu, S. & Valavaara, T. Demonstration environment for smart grid applications. [PDF]. Second version. Tampere University of Technology, Department of Electrical Energy Engineering. 26.4.2011.

[120] Robinson, G. & Zhou, M.J. Utility applications should be integrated with an interface based on a canonical data model, not directly with each other. Power Systems Conference and Exposition, 2004. IEEE PES, 2004, pp. 1592-1596 vol.3.

[121] Rosanova, D. Microsoft BizTalk Server 2010 Patterns. Birmingham, UK 2011, Packt Publishing. 371 p.

[122] Seroter, R., Fairweather, E., Thomas, S.W., Sexton, M. & Ramani, R. Applied Architecture Patterns on the Microsoft Platform. Birmingham, UK 2010, Packt Publishing. 518 p.

[123] Sierla, S. Middleware Solutions for Automation Applications - Case RTPS. Master's thesis. Espoo 2003. Helsinki University of Technology. Department of Automation and Systems Technology, Information and Computer Systems in Automation Report 9. 96 p.

[124] Smart Grid Interoperability Panel Cyber Security Working Group. Guidelines for Smart Grid Cyber Security: Vol. 1, Smart Grid Cyber Security Strategy, Architecture, and High-Level Requirements. 2010, NIST, National Institute of Standards and Technology Interagency Report 7628, vol. 1. 289 p.

[125] Smart Grid Interoperability Panel Cyber Security Working Group. Guidelines for Smart Grid Cyber Security: Vol. 2, Privacy and the Smart Grid. 2010, NIST, National Institute of Standards and Technology Interagency Report 7628 vol. 2. 69 p.

[126] Smart Grid Interoperability Panel Cyber Security Working Group. Guidelines for Smart Grid Cyber Security: Vol. 3, Supportive Analyses and References. 2010, NIST, National Institute of Standards and Technology Interagency Report 7628 vol. 3. 219 p.

[127] Smart Grid Interoperability Panel Cyber Security Working Group. Introduction to NISTIR 7628 Guidelines for Smart Grid Cyber Security. 2010, NIST, 20 p.

[128] SmartGrids European Technology Platform. The SmartGrids European Technology Platform. [WWW]. [Referred: 19.2.2013]. Available at: http://www.smartgrids.eu/ETPSmartGrids.

[129] Stallings, W. Cryptography and Network Security. 1999, Prentice Hall.

[130] Stallings, W. Cryptography and Network Security - Principles and Practice. 5th edition. USA 2011, Pearson Education. 719 p.

[131] Stouffer, K., Falco, J. & Scarfone, K. Guide to Industrial Control System (ICS) Security. 2011, National Institute of Standards and Technology, Recommendations of the National Institute of Standards and Technology NIST Special Publication 800-82. 155 p.

[132] Structured Analysis Wiki. Chapter 9 - Dataflow Diagrams. [WWW]. 13.6.2011. [Referred: 24.10.2012]. Available at: http://yourdon.com/strucanalysis/wiki/index.php?title=Chapter_9.

[133] Suomen Automaatioseura. Teollisuusautomaation tietoturva: verkottumisen riskit ja niiden hallinta. [PDF]. 1. verkkopainos. Suomen Automaatioseura Ry,

Turvallisuusjaosto. 2010. [Retrieved: 7.6.2012]. Available: http://www.cert.fi/attachments/cip/5na1SblCp/SAS29_TeollisuusautomaationTietoturva.pdf.

[134] Söder, L. Presentation in SGEM Unconference. Kirkkonummi, Finland 6.9.2012.

[135] TEKES. Strategic Centres for Science, Technology and Innovation. [WWW]. 2011. [Referred: 12.6.2012]. Available at: http://www.shok.fi/en/.

[136] Thornton A. & Monroy C.R. Distributed power generation in the United States. Renewable and Sustainable Energy Reviews 15(2011)9, pp. 4809-4817.

[137] Toivonen, J., Trygg, P., Antila, S., Mäkinen, A., Järventausta, P., Mäenpää, T., Nyrhilä, V., Saaristo, H. & Mattsson, J. Sähköyhtiöiden tietojärjestelmä-kartoitus. Tampere 2005, Tampereen teknillinen yliopisto, Sähkövoima-tekniikan laitos 2005:2. 56 p.

[138] Toivonen, J., Antila, S. & Järventausta, P. XML tietojärjestelmien integroin-nissa - case: kuntotarkastussovellus. Tampere 2006, Tampereen teknillinen yliopisto, Sähkövoimatekniikan laitos 2006:4. 33 s.

[139] Troy, R. & Helmke, M. VMware Cookbook. 2nd edition. Sebastopol, CA 2012, O'Reilly Media. 341 p.

[140] UCA International Users Group. CIM User Group. [WWW]. [Referred: 13.8.2012]. Available at: http://cimug.ucaiug.org/default.aspx.

[141] Ueno, K. & Tatsubori, M. Early Capacity Testing of an Enterprise Service Bus. ICWS '06. Proceedings of International Conference on Web Services, 2006. pp. 709-716.

[142] Uslar, M., Schmedes, T., Lucks, A., Luhmann, T., Winkels, L. & Appelrath, H. Interaction of EMS related systems by using the CIM standard. [PDF]. University of Oldenburg. [Retrieved: 4.11.2012]. Available: http://www-is.informatik.uni-oldenburg.de/publications/1881.pdf.

[143] VMware. Benefits of virtualization, Increase IT efficiency and virtual management. [WWW]. 2013. [Referred: 17.1.2013]. Available at: http://www.vmware.com/virtualization/.

[144] VMware. How Virtualization Works? [WWW]. 2013. [Referred: 9.3.2013]. Available at: http://www.vmware.com/virtualization/virtualization-basics/how-virtualization-works.html.

[145] Verho, P., Kärenlampi, M., Pitkänen, M., Järventausta, P. & Partanen, J. Distribution Management System. Tampere 1997, Tampere University of Technology, Power Engineering 5-97. 82 p.

[146] Wang J. & Rong L. Cascade-based attack vulnerability on the US power grid. Safety Science 47(2009)10, pp. 1332-1336.

[147] Weiss, J.P. Assuring Industrial Control System (ICS) Cyber Security. [PDF]. 25.8.2008. [Retrieved: 19.8.2012]. Available: http://csis.org/files/media/csis/pubs/080825_cyber.pdf.

[148] Werner, T., Vetter, C., Kostic, T. & Lohmann, V. Data exchange in asset management applications for electric utilities using XML. APSCOM-00. Proceedings of the 2000 International Conference on Advances in Power System Control, Operation and Management, 2000. pp. 220-224 vol.1.

[149] World Wide Web Consortium, W3C. All Standards and Drafts. [WWW]. [Referred: 16.9.2012]. Available at: http://www.w3.org/TR/.

[150] World Wide Web Consortium, W3C. World Wide Web Consortium. [WWW]. [Referred: 18.9.2012]. Available at: http://www.w3.org/.

APPENDIX A

Table A.1. *Future vs. today's grid, combined from [32;42, p.514;74;100].*

Comparison between today's grid and the Smart Grid		
Goal	Current grid	Smart Grid
Self-heals	Responds to prevent further damage. Focus is on protection of assets following system faults.	Automatically detects and responds to actual and emerging transmission and distribution problems. Focus is on prevention. Minimizes consumer impact.
Motivates and includes the consumer	Consumers are uninformed and non-participative with the power system.	Informed, involved and active consumers. Broad penetration of demand response.
Resists attacks and disasters	Vulnerable to malicious acts of terror and natural disasters.	Resilient to attack and natural disasters with rapid restoration capabilities.
Provides power quality (PQ) for twenty-first century needs	Focused on outages rather than power quality problems. Slow response in resolving PQ issues.	Quality of power meets industry standards and consumer needs. PQ issues identified and resolved prior to manifestation. Various levels of PQ at varying prices.
Power flow	One-directional power flow.	Controllable, multi-directional power flow.
Real-time operations on all levels	Operation based on historical experience (MV-LV distribution networks)	Operation based on real-time data.
Accommodates all generation and storage options	Relatively small number of large generating plants. Numerous obstacles exist for interconnecting distributed energy resources.	Very large numbers of diverse distributed generation and storage devices deployed to complement the large generating plants. Plug-and-play convenience. Significantly more focus on and access to renewables.
Enables and improves markets, both wholesale and small-scale Distributed Generation (DG) markets	Limited wholesale markets still working to find the best operating models. Not well integrated with each other. Transmission congestion separates the buyers and sellers. Weak market integration for DG.	Mature wholesale market operations in place; well integrated nationwide and integrated with reliability coordinators. Retail markets flourishing where appropriate. Minimal transmission congestion and constraints. DERs are integrated into energy market and power systems.
Optimises assets and operates efficiently.	Minimal integration of limited operational data with asset management processes and technologies. Siloed business processes. Time based maintenance.	Greatly expanded sensing and measurement of grid conditions. Grid technologies deeply integrated with asset management processes for the most effective management of assets and costs. Condition based maintenance.

APPENDIX B

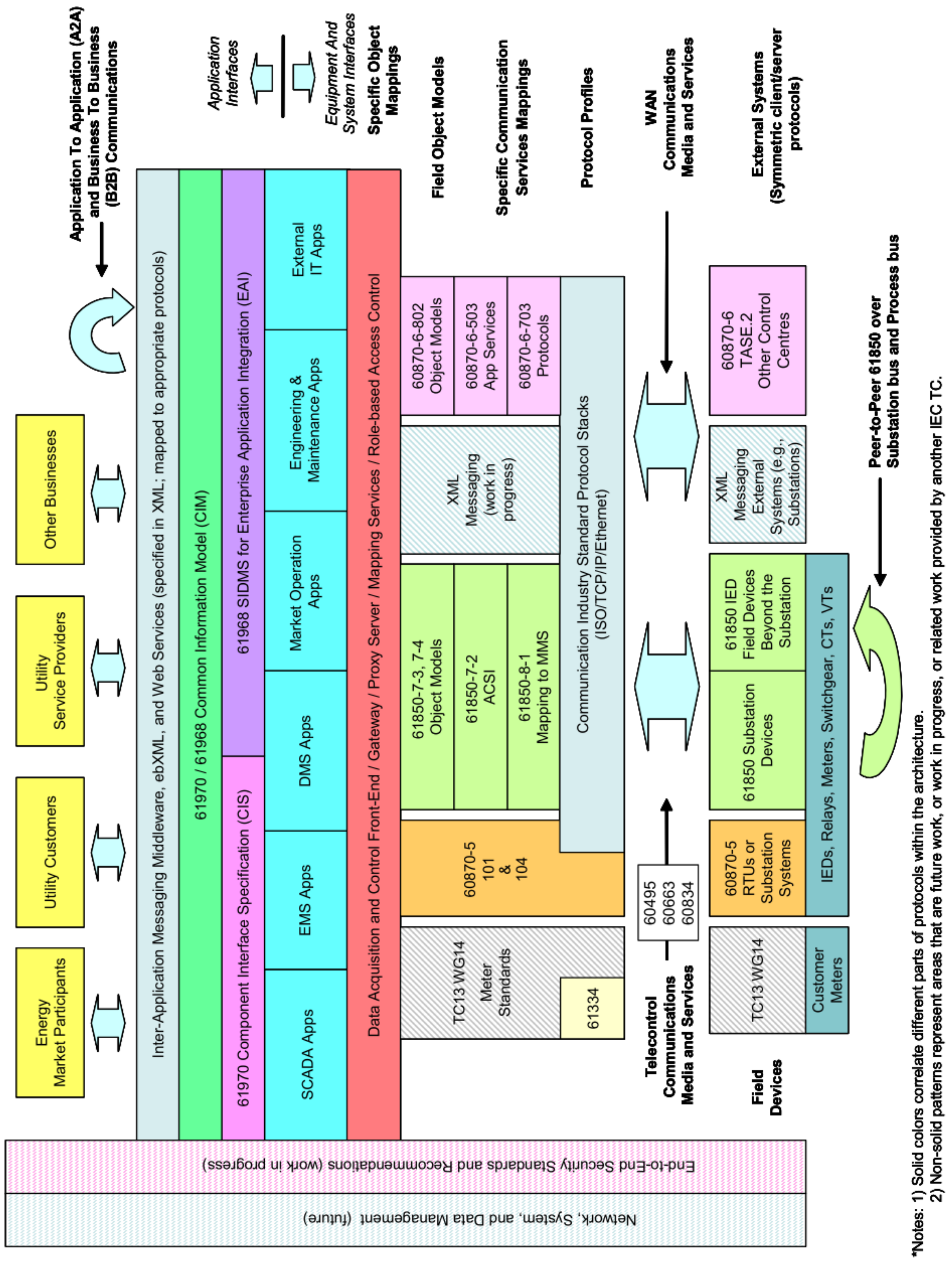
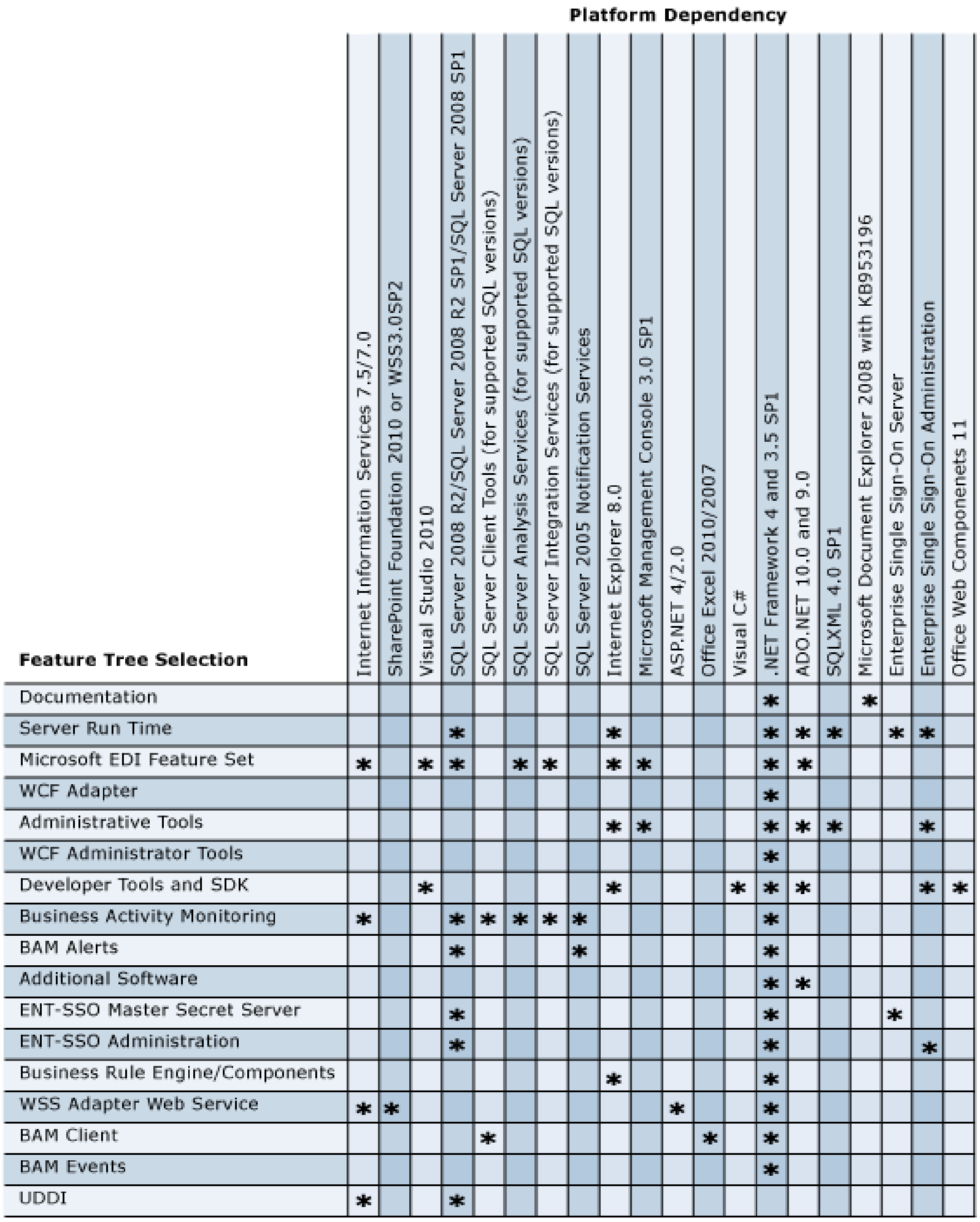

Energy Market Participants
Utility Customers
Utility Service Providers
Other Businesses
Application To Application (A2A) and Business To Business (B2B) Communications
Application Interfaces
Equipment And System Interfaces
Specific Object Mappings
Field Object Models
Specific Communication Services Mappings
Protocol Profiles
WAN Communications Media and Services
External Systems (Symmetric client/server protocols)
Inter-Application Messaging Middleware, ebXML, and Web Services (specified in XML; mapped to appropriate protocols)
61970 / 61968 Common Information Model (CIM)
61970 Component Interface Specification (CIS)
61968 SIDMS for Enterprise Application Integration (EAI)
SCADA Apps
EMS Apps
DMS Apps
Market Operation Apps
Engineering & Maintenance Apps
External IT Apps
Data Acquisition and Control Front-End / Gateway / Proxy Server / Mapping Services / Role-based Access Control
TC13 WG14 Meter Standards
61334
60870-5 101 & 104
61850-7-3, 7-4 Object Models
61850-7-2 ACSI
61850-8-1 Mapping to MMS
XML Messaging (work in progress)
60870-6-802 Object Models
60870-6-503 App Services
60870-6-703 Protocols
Communication Industry Standard Protocol Stacks (ISO/TCP/IP/Ethernet)
Telecontrol Communications Media and Services
60495
60663
60834
Field Devices
TC13 WG14
60870-5 RTUs or Substation Systems
61850 Substation Devices
61850 IED Field Devices Beyond the Substation
XML Messaging External Systems (e.g., Substations)
60870-6 TASE.2 Other Control Centres
Customer Meters
IEDs, Relays, Meters, Switchgear, CTs, VTs
Peer-to-Peer 61850 over Substation bus and Process bus
End-to-End Security Standards and Recommendations (work in progress)
Network, System, and Data Management (future)
*Notes: 1) Solid colors correlate different parts of protocols within the architecture.
2) Non-solid patterns represent areas that are future work, or work in progress, or related work provided by another IEC TC.

Figure B.1. *The IEC Reference architecture [70;71].*

APPENDIX C



	Platform Dependency																			
Feature Tree Selection	Internet Information Services 7.5/7.0	SharePoint Foundation 2010 or WSS3.0SP2	Visual Studio 2010	SQL Server 2008 R2 SP1/SQL Server 2008 SP1	SQL Server Client Tools (for supported SQL versions)	SQL Server Analysis Services (for supported SQL versions)	SQL Server Integration Services (for supported SQL versions)	SQL Server 2005 Notification Services	Internet Explorer 8.0	Microsoft Management Console 3.0 SP1	ASP.NET 4/2.0	Office Excel 2010/2007	Visual C#	.NET Framework 4 and 3.5 SP1	ADO.NET 10.0 and 9.0	SQLXML 4.0 SP1	Microsoft Document Explorer 2008 with KB953196	Enterprise Single Sign-On Server	Enterprise Single Sign-On Administration	Office Web Componenets 11
Documentation														*			*			
Server Run Time				*					*					*	*	*		*	*	
Microsoft EDI Feature Set	*		*	*		*	*		*	*				*	*					
WCF Adapter														*						
Administrative Tools									*	*				*	*	*			*	
WCF Administrator Tools														*						
Developer Tools and SDK			*						*				*	*	*				*	*
Business Activity Monitoring	*			*	*	*	*	*						*						
BAM Alerts				*				*						*						
Additional Software														*	*					
ENT-SSO Master Secret Server				*										*				*		
ENT-SSO Administration				*										*					*	
Business Rule Engine/Components									*					*						
WSS Adapter Web Service	*	*									*			*						
BAM Client					*							*		*						
BAM Events														*						
UDDI	*			*																

Figure C.1. *BizTalk 2010 features and core dependencies [94].*

APPENDIX D

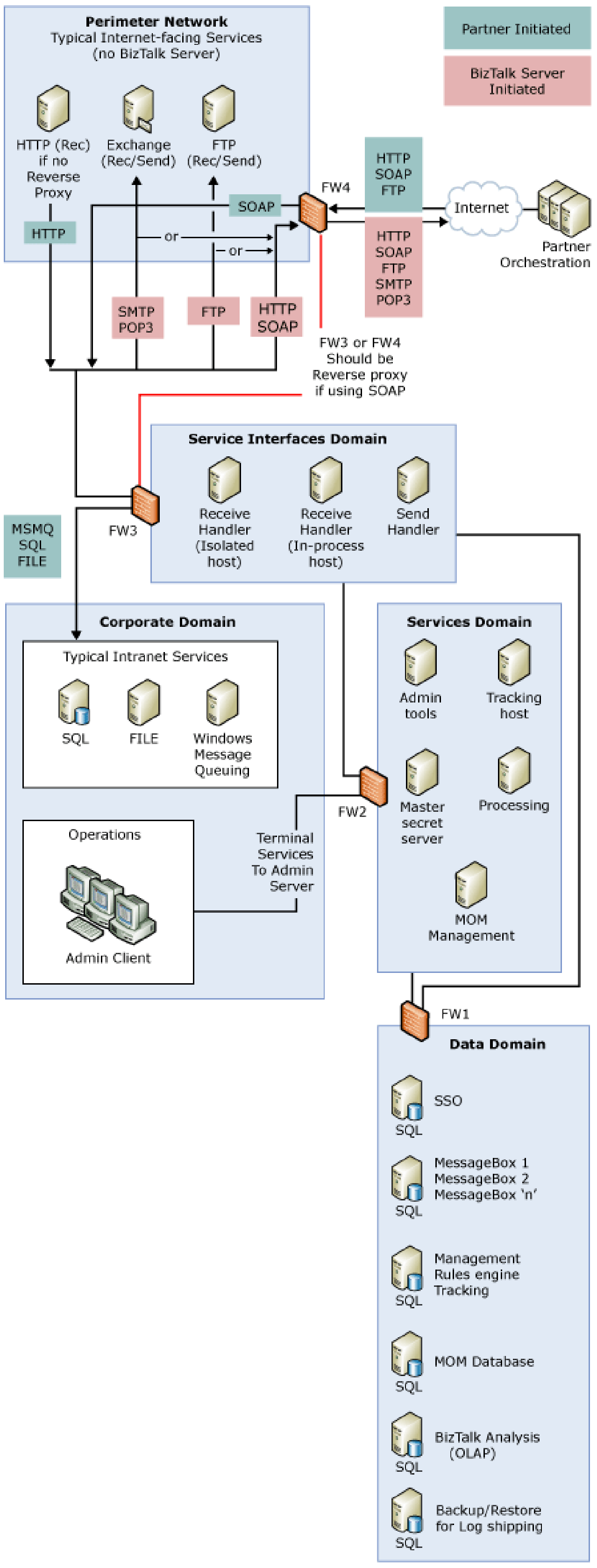


Figure D.1. *Highly distributed BizTalk architecture with defence-in-depth [96].*