LAPPEENRANTA UNIVERSITY OF TECHNOLOGY

Department of Information Technology

Master of Science Thesis

QUALITY STANDARDS IN E-BUSINESS SOFTWARE DEVELOPMENT

Supervisors: Professor Kari Smolander

D.Sc. (Tech) Ossi Taipale

Lappeenranta, 15th April, 2009

Tero Pesonen

Linnunrata 10 H 4

53850 Lappeenranta

tero.pesonen@lut.fi

ABSTRACT

Author: Tero Pesonen

Title: Quality standards in e-business software development

Department: Information technology

Year: 2009

Master's thesis. Lappeenranta University of Technology.

97 pages, 16 figures and 4 tables

Supervisors: Professor Kari Smolander

D.Sc. (Tech) Ossi Taipale

Keywords: Standards, software testing, quality assurance, eBusiness, B2Bi.

Large enterprises have for many years employed eBusiness solutions in order to improve their efficiency. Smaller companies, however, have not been able to leverage these technologies due to the high level of know-how and resources required in implementing them. To solve this, novel software services are being developed to facilitate eBusiness adoption for the small enterprise with the aim of making B2Bi feasible not only between large organisations but also between trading partners of all sizes. The objective of this study was to find what standards and techniques on eBusiness and software testing and quality assurance fit best for building these new kinds of software considering the requirements their unique eBusiness approach poses. The research was conducted as a literature study with focus on standards on software testing and quality assurance together with standards on eBusiness.

The study showed that the current software testing and quality assurance standards do not possess such characteristics as would make select standards evidently better fitted for building this type of software, which were established to be best developed as web services in order for them to meet their requirements. A selection of eBusiness standards and technologies was proposed to support this approach. The main finding in the study was, however, that these kinds of web services that have high interoperability requirements will have to be able to carry out automated interoperability and conformance testing as part of their operation; this objective dictates how the software are built and how testing during software development is to be done. The study showed that research on automated interoperability and conformance testing for web services is still limited and more research is needed to make the building of highly-interoperable web services more feasible.

TIIVISTELMÄ

Tekijä: Tero Pesonen

Nimi: Quality standards in e-business software development

Osasto: Tietotekniikka

Vuosi: 2009

Diplomityö. Lappeenrannan teknillinen yliopisto.

97 sivua, 16 kuvaa ja 4 taulukkoa.

Tarkastajat: Professori Kari Smolander

TkT Ossi Taipale

Hakusanat: Standardit, ohjelmistotestaus, laadunvarmistus, e-liiketoiminta.

Suuryritykset ovat jo pitkään käyttäneet e-liiketoimintaan perustuvia ratkaisuja kilpailukykynsä parantamiseen. Pk-yritykset sen sijaan ovat jääneet tästä kehityksestä paitsi, sillä niiltä puuttuvat tällaisten ratkaisujen käyttöönottoon tarvittavat resurssit ja asiantuntemus. Jotta myös pienemmät toimijat voisivat hyötyä liiketoimintaprosessien sähköistämisestä, on ratkaisuksi esitetty uudenlaisia ohjelmistotuotteita joiden toteuttamat internetpalvelut mahdollistavat e-liiketomintamenetelmien käyttöönoton pk-yrityksissä siten että myös suurten organisaatioiden ja niiden pk-kumpanien välinen sanomaliikenne voidaan sähköistää. Tutkimuksen tavoitteena oli selvittää, mitkä ohjelmistotestauksen ja laadunhallinnan, sekä toisaalta e-liiketoiminnan standardit ja menetelmät soveltuvat parhaiten tällaisten uudenlaisten ohjelmistotuotteiden kehitystyöhön. Olettamuksena oli, että ohjelmistojen uudenlainen tapa hyödyntää e-liiketoiminnan menetelmiä asettaa myös uudenlaisia vaatimuksia ohjelmistokehitysprojektille.

Tutkimuksessa ei havaittu merkityksellisiä eroja testauksen ja laadunhallinan standardien välillä suhteessa ohjelmistoista löydettyihin erityisvaatimuksiin. Sen sijaan tutkimus osoitti että korkean yhteensopivuustarpeen omaavien web service ohjelmistojen on voitava suorittaa automatisoitua yhteensopivuus- ja yhdenmukaisuustestausta jotta ne voivat tehokkaasti hyödyntää e-liiketoimintamenetelmiä ja toteuttaa näihin nojaavan toiminnallisuuden. Vaatimus kyvystä automatisoituun ajonaikaiseen testaukseen määrittää ohjelmistokehityksen aikana tehtäviä valintoja ja siten myös käytettävät e-liiketoiminnan standardit ja tekniikat. Kartoitettaessa automatisoidun web service yhteensopivuustestauksen menetelmiä havaittiin, että tutkimus on tällä saralla ollut vasta vähäistä. Lisätutkimusta tarvitaan jotta uudenlaisten e-liiketoiminnan web service ohjelmistojen kehitys helpottuisi.

FOREWORD

This master's thesis was written as part of MASTO research project at Lappeenranta University of Technology during a six-month period from October 2008 to April 2009. In order to get acquainted with the research topic, a short work period at the partner company Lappeenranta Innovation Ltd was arranged in October and November 2008. This was a great experience, and I would like to use the opportunity to thank everyone at Inno for making me feel very welcome during my stay. I would especially like to thank Kari Korpela for sharing his expertise on eBusiness topics and for helping me with many comments and suggestions along the way. It has been a great experience to work with you.

I also owe special thanks to my instructors Dr. Ossi Taipale and Prof. Kari Smolander for all the help and advice I have received during my writing this thesis. I also want to thank you for the opportunity to work at TBRC and in MASTO. I have learnt a lot, and am thankful for your encouragement to pursue post-graduate studies.

Lopuksi haluan vielä kiittää vanhempiani ja veljeäni saamastani tuesta ja kannustuksesta.

Table of Contents

ABBREVIATIONS

API	Application Programming Interface
B2Bi	Business-to-business integration
B2G	Business-to-government
BPSS	Business Process Specification Schema
CC	Core component
CEN	The European Committee for Standardization
CPA	Collaboration Protocol Agreement
ebMS	ebXML Message Service
eBusiness	Electronic business
ebXML	Electronic business using eXtensible markup language
CPP	Collaboration Profile Protocol
EDI	Electronic Data Interchange
EDIFACT	Electronic Data Interchange for Administration Commerce and Transport
eGovernance	Electronic governance
DTD	Document Type Definition
ERP	Enterprise resource planning
HTML	Hyper Text Markup Language
ICT	Information and communication technology
IEC	International Electrotechnical Commission
IDL	Interface Definition Language
ISO	International Standards Organization
ITU	International Telecommunication Union

MEP	Message exchange pattern
MNC	Multinational company
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
OWL	Web Ontology Language
PDF	Portable Document Format
PIP	Partner Interface Process
QA	Quality assurance
RAE	RosettaNet Automated Enablement
RDF	Resource Description Framework
RNF	RosettaNet Implementation Framework
RPC	Remote Procedure Call
SME	Small and medium enterprise
SOAP	Simple Object Access Protocol
TRIP-PIP	Trading Partner Implementation Requirements PIP
TRIP-PF	Trading Partner Implementation Requirements Presentation Format
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Service Definition Language
XML	Extensible Markup Language

1. INTRODUCTION

The aim at using electronic business (eBusiness) in the enterprise is to improve efficiency. eBusiness is not new: For over two decades, large, multinational companies (MNC's) have been using such eBusiness standards as EDI to reduce their clerical costs by eliminating many kinds of manual work and related cost factors. With newer RosettaNet-like eBusiness frameworks, seamless business process integration between trading partners is becoming almost standard in some industries – the digitalisation of business processes is changing how business is conducted. However, although MNC's have led the way in leveraging these new technologies, small and mid size enterprises (SME's) have not been able to follow suit. The heavy eBusiness frameworks, developed for the needs posed by MNC's, require such know-how and resources as a typical SME does not have. The situation where SME's cannot benefit from this development affects not only their ability to conduct business efficiently but also all MNC-to-SME and government-to-SME electronic data exchange. The objective of the case projects subject to study in this thesis is to remove the key barriers now present in setting up these kinds of B2Bi.

This thesis was done as part of a software testing and quality assurance research project MASTO at Lappeenranta University of Technology. The research topic for the thesis became two-fold in the sense that the issue of quality in the development of the case projects was approached from two different angles: On one hand, in the form of standards on testing and quality assurance; on the other hand, in the form of standards and technologies on eBusiness. The research question became, thus: What standards and techniques on eBusiness and software testing and quality assurance would best fit for the development of the eLive case software projects, considering their novel approach for building software for B2Bi enablement.

The research question has a wide theoretical scope. First, it is concerned with the general issues on eBusiness together with the particulars on applying eBusiness standards. It also requires the understanding of testing and quality assurance for software development so that standards on testing and eBusiness can be studied and

evaluated against the software-development-oriented requirements perceived in the case projects.

The research was conducted as a literature study by which an answer to the research question could be derived. The thesis assumed the following structure: The first three chapters on eBusiness, standards and standardisation, and testing and quality assurance in software development, respectively, provide the theoretical framework for the last two, longer chapters, where testing and quality assurance standards (chapter 5) and eBusiness standards and techniques (chapter 6) are studied; chapter 6 also describes the case projects. Finally, chapter 7 provides the results obtained in the study.

2. ELECTRONIC BUSINESS

Electronic business, or electronic commerce, commonly referred to as eBusiness or eCommerce for short, involves utilising information and communication technology (ICT) in support of business activities. The term electronic commerce is sometimes used more strictly to refer to such eBusiness activities as consist of buying and selling of goods and services, while the more general eBusiness term is then reserved for any kind of business or business-related activity conducted between individuals and/or organisations so that data are processed electronically. Hence, a typical venue for modern eBusiness is the internet, with its applications varying from on-line shopping, business-to-business (B2B) data exchange (orders, invoices, etc.) all the way to electronic governance. (Voutilainen & Pento, 2003; Andam, 2003)

Even though eBusiness has existed alongside the conventional, non-electronic forms of business and data exchange in most industrialised nations since the very early days of electronic banking systems in the 1970s, it was not until the rapid development and adoption of ICT in the mid 1980s to mid 1990s that electronic business as it is conceived today became defined. Furthermore, with the advent of the internet and the rapid growth in its adoption for business transactions, eBusiness merged with the wider concept of information society to become a tool for the envisaged benefits in networking (and computerising) homes, organisations, and finally, whole societies. (Vuorensyrjä & Savolainen, 2001)

Despite being linked with ICT, the growth in eBusiness has not traditionally, nor does it today, follow strictly the adoption of ICT. Even within economically more homogeneous regions, such as the European Union (EU), the adoption rate of eBusiness by both individuals and small to mid size enterprises (SME's) remains more uneven than the use of ICT in general. Yet, today eBusiness applications are seen ever more important in realising the economical benefits in ICT investments. Therefore, local and global research on eBusiness adoption and its drivers and obstacles has increased rapidly. (Fillis et al. 2004; Pulli 2005; Bogatar & Pucihar 2007)

2.1 eBusiness applications

The first widely-deployed eBusiness applications were brought into use in 1980s by large and multinational companies (MNC's) who established to improve their efficiency by implementing methods for electronic exchange of business documents in order to lower transaction costs and to reduce errors. (Phan & sommer 1999; Voutilainen & Pento, 2003) For this purpose the standards UN/EDIFACT (United Nations / Electronic Data Interchange for Administration Commerce and Transport), ASC X 2 and EANCOM, were developed.

While many MNC's and their larger business partners adopted the EDI standard, it has seen little use in middle-sized companies and has been adopted by only select small businesses. This was due to the complexity and costs involved in implementing EDI – often beyond the capabilities of even middle-sized companies – and EDI, together with its most noticeable follower in the 1990s, RosettaNet, have been implemented in smaller businesses only when their business transactions with an MNC have specifically required it. As such, the situation where only large enterprises benefit from these eBusiness applications remains today much unchanged from the late eighties. (Fillis et al. 2004; Bogatar & Pucihar 2007; Korpela et al. 2007)

Consumer and SME-centric eBusiness applications arrived with the growth of ICT and the resultant advent of the internet in the mid to late 1990s, both of which brought electronic data exchange to the reach of the consumer and the SME. At first, these applications included only rudimentary solutions that enabled the consumer to buy products on-line, or, for example, to access such information as is related to buying and selling goods. The early eBusiness applications were only loosely (if at all) connected to the back-end systems of the organisations they served and rather lived apart as mere add-ons to the traditional business model. (Vuorensyrjä & Savolainen, 2001)

In the next phase of eBusiness development, from late nineties onwards, both B2B and consumer and SME related eBusiness grew ever more integrated with the organisations that applied them. Wholly new internet-only (operating only by means of eBusiness) businesses were conceived, and such monikers as *eGovernance* stepped forth. The latter

came to mean such services as provided by governing bodies to citizens and businesses, as well as other organisations within the government or jurisdiction, where the traditional one- or two-way communication, managed by fill-out forms and visiting bureaus in person, could be conducted via the internet by on-line forms or by accessing web sites, or by other electronic means. On-line voting serves as an example of the latest eGovernance initiatives. (Shailendra et al. 2008)

It should be noted, however, as pointed out earlier, that although no such complexity or cost-related constraints exist in implementing the aforementioned-like SME-level eBusiness solution as do with MNC-centric B2B standards, the adoption and usage rates of even the simplest forms of eBusiness vary greatly globally and even locally, within nations. (Fillis et al. 2004) The obstacles, as well as the drivers, for eBusiness adoption in this sector are hence vastly different to those in the MNC space. As noted by Jantavongso & Sugianto (2006), the benefits SME's perceive in adopting eBusiness applications vary between companies not only from different regions but in larger term from developed and developing nations; this is also true for the perceived threats and business-risks associated with implementing modern eBusiness applications.

Where the latest phase in eBusiness development has meant "making all things on-line" for the consumer and the small business, in the MNC and large enterprise space there has been notable development not only on that score but also in the very areas that started the eBusiness development in the eighties. While EDI and its early counterparts were able to make B2B data exchange digital, more automated, and less error-prone, modern enterprise-geared B2B solutions aim at tackling an even vaster range of costs-reduction issues. These include – in the aim at further reducing clerical costs – integrating eBusiness platforms seamlessly with enterprise resource planning (ERP) systems and upgrading and automating document handling by employing XML (Extensible Markup Language) -based standards in place of the older EDIFACT language. The trend is in leveraging eBusiness related standards to build ever more integrated solutions.

In addition to building upon existing eBusiness models, national and international standards and practises have been developed for facilitating cross-border trade

(eCustoms, eInvoices) and security and authentication issues in B2B and B2G (business to government) electronic data exchange, thereby creating new ways for using eBusiness to improve operational efficiency. The problem of how to have small organisations to also benefit from these developments in the "heavy", MNC-level eBusiness applications, has lately received more attention. (Mykkänen et al. 2005; Boh et al. 2007; Korpela et al. 2007)

2.2 eBusiness at the SME level

The focus on the case eBusiness projects studied in this thesis – eYellowpages and eCatalogue – lies in the sphere of inter-SME, SME-to-MNC and SME-to-government eBusiness enablement. Therefore, a short overview is provided on how SME's today make use of eBusiness and what factors affect eBusiness adoption, and how implementing electronic business models affects their ability to increase efficiency or to create new business.

2.2.1 eBusiness adoption by SME's

Although the economical environment in which SME's operate varies greatly, most of the motifs for adopting or assessing possible adoption of eBusiness solutions are rather universal – as shown by the studies referenced below – and include in general the desire to improve efficiency, to gain competitive edge or to facilitate growth into new markets.

In the developed world, aforementioned-like motives have been expected to drive SME eBusiness adoption "on their own," as it has seemed only natural that businesses adopt more efficient ways to operate or new ways to grow, and on the other hand, would adapt to the changes in the competitive environment by investing in eBusiness technologies should their competition do so as well.

Despite western SME's generally higher readiness (known as e-readiness, i.e. ability) in adopting ICT-based solutions, this development has not taken place on the scale predicted, even if it is quite true that SME's do perceive benefits in implementing eBusiness solutions; the awareness of which has in part been helped by many western

governments' attempts at increasing eBusiness adoption amongst the SME sector, supported by research in the field. (Mansikka 2002; Murphy & Taylor 2004; Beheshti & Salehi-Sangari 2007)

Studies in general seem to agree that this has a lot to do with how the way SME's operate differs from larger companies: While both share some common goals, such as long term profitability, etc., that could be expected to promote the adoption of more efficient methodologies or wholly new approaches to making business – as is evident in the creation of such eBusiness standards for MNC's as EDI – the SME is constrained in trying to achieve this by limited resources and, as emphasised by Fillis et al. (2004), often by the impact of an owner or a manager exerting a high degree of control in decision making. In other words, SME's work very differently to MNC's in adopting and adapting to new, and for reasons that may not relate at all to actual business goals.

It seems that in the developing world, on the other hand, SME eBusiness awareness (and, hence, adoption) has been more driven by the markets than in the west: eBusiness technologies are seen as key to levelling the playing field with larger companies, or in expanding abroad, or, for example, in improving service quality; all of which are obvious business goals. The need to exchange data with their trading partners in an electronic form is one driver both western and non-western SME's share. (Deschoolmeester et al. 2004; Jantavongso & Sugianto 2006; Du et al. 2007)

The rate in which SME's have adopted eBusiness technologies has varied tremendously, both globally and within countries or even smaller regions. (E-business Policy Group 2002; Bogatar & Pucihar 2007) It is important to understand, however, that the concepts *eBusiness* and *SME* are understood somewhat differently by researchers in different countries, so strict one-to-one comparisons are not always meaningful or even feasible.

2.2.2 eBusiness drivers and barriers

The findings of Du et al. (2007), and Jantavongso & Sugianto (2006), who have studied eBusiness adoption in China and Thailand, respectively, conform with those of similar studies in West (e.g. Fillis et al. 2004 and Deschoolmeester et al. 2004), in that there

exists no specific, universal eBusiness driver or barrier that would apply to the general SME; rather, the issues of eBusiness adoption are industry, culture and country, region, and above all, company specific. In other words, although there exist certain trends amongst western or non-western SME's, or SME's in a given country or region, the differences between the exact circumstances surrounding a particular SME are versatile; this does not, however, preclude broad-level generalisations: eBusiness development within a country or region can be described by a number of factors and thus also be predicted, as shown by Mansikka (2002) and similar earlier-referenced studies in eBusiness up-take in regional contexts.

The kinds of barrier that have been found to affect SME's to various extends, depending on the study context, are issues on costs, changes of corporate culture, security, resources, resistance to change by employees, and in less e-ready countries, issues on education. Also, a trend that does emerge in a number of studies is that when it comes to SME-to-MNC eBusiness interaction, the obstacles in implementing the MNC-imposed eBusiness practises at the SME-level are uniform around the globe. (Beheshti & Salehi-Sangari 2004; Korpela et al. 2007; Wah et al. 2007)

2.3 eBusiness as an innovation

The spread and adoption of eBusiness at MNC, SME or consumer level may in addition to the idea of specific drivers and barriers be approached from the concept of innovation and how innovations spread. In this context, eBusiness in general, or a given eBusiness application or application set, is viewed as an *innovation,* defined by Rogers (1995) as "*an idea, practice, or object that is perceived as new by an individual or other unit of adoption* ."

Technological innovations are further defined as having two intrinsic aspects: Software and hardware (properties), the former of which has to do with the information base for the tool, while the latter affects how the technology is embodied as a physical object. These attributes are often spoken by such more lax terms as "computer software", "code" and "computer hardware" or "semiconductors." Although most technological

innovations hold to a certain degree both of these aspects, typical eBusiness innovations live more on the software side. (Rogers 1995; Mansikka 2002)

In this work, as in similar earlier studies (Mansikka 2002, Pulli 2005) eBusiness applications are treated as technological innovations, although some eBusiness applications may possess qualities often associated with social innovations, the aim of whom lies in solving issues present in the larger concept of human society, and which hence go beyond what is defined as technology.

2.3.1 Diffusion of innovations

The dissemination and acceptance of innovations in the markets or amongst parties is often called diffusion. This process is depicted in figure 1 on page 13. According to Rogers & Scott (1997), the diffusion process involves (1) an innovation, which is communicated through certain (2) channels over (3) time (4) amongst the members of a social system. Diffusions can be understood as a special kind of communication concerned with the spread of messages that are perceived as new ideas, i.e. innovations.

As figure 1 shows, not all innovations are adopted the same way. The rate in which an innovation spreads is highly dependent on how the members of the social system perceive its characteristics. These characteristics include: (1) Relative advantage, which is the perceived supremacy of the innovation compared to the one it supersedes; (2) compatibility, which measures how well the innovation is perceived to comply with the values, past experiences and needs of its potential adopters; (3) complexity, which expresses how difficult to use or understand the innovation is perceived; (4) trialability, measuring the degree to which the innovation may be experimented with on a limited bases; and (5) observability, the measure to which the results of the innovation are visible to others.

The process by which the innovations spread is communication, for which communication channels form the basic means. Such channels may include, for instance, mass media or peer-to-peer exchange of information. Communication channels are very important in determining how an individual perceives the innovation;

much more so than, for example, scientific research or purely factual information, as the subjective evaluations of near-peers typically outweigh other sources of influence.

Innovations spread in social systems, which are defined as inter-related units that engage in joint problem solving in order to accomplish a common goal. This means that the "units" of which the system comprises may be anything from individuals to large organisations or even social subsystems. Together with the boundaries of the social system time is a major determiner in how the diffusion may take place. This is depicted in the form of the S-curves in figure 1. (Rogers 1995; Rogers & Scott 1997; Pulli 2005)

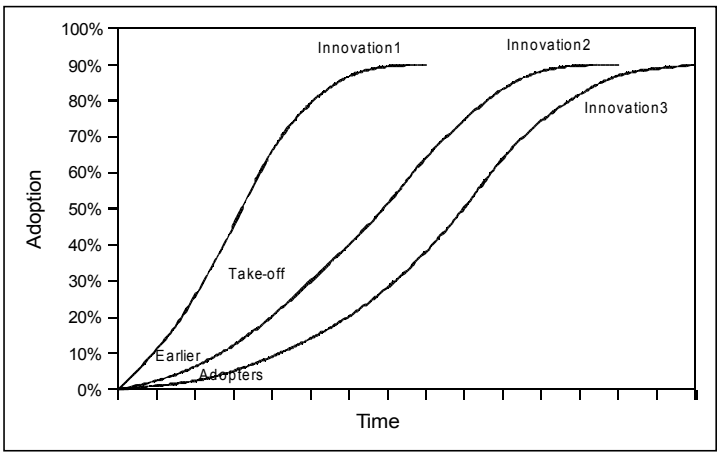

Figure 1: The diffusion of innovations. (Rogers & Scott 1997)

2.4 Summary

Electronic business (eBusiness) involves utilising information and communication technology (ICT) in support of business activities. eBusiness applications are concerned with electronic representation of business data and processes, which allow organisations to integrate and exchange data more efficiently. eBusiness has grown with the adoption of ICT, but not as fast or as universally – eBusienss adoption is highest amongst MNC's, as they have had the technological expertise as well as the resources to build systems that may use EDI- or RosettaNet-like eBusiness frameworks, which help to decrease clerical costs and improve efficiency. This has not been equally possible for SME's, amongst which eBusines adoption of any kind varies significantly even within region-, culture-, or industry-wise homogeneous environments.

Studies show that most SME's lack the resources as well as the know-how to capitalise on the eBusiness development now on-going. It is also true that SME's approach their business goals differently to MNC's and perceive eBusiness benefits and risks differently. Although there are visible trends in SME eBusiness adoption, the exact drivers and barriers most affecting a given SME cannot be reliably derived from these studies given how versatile the circumstances for SME's are. A trend that does, however, emerge is that the obstacles in implementing the MNC-imposed eBusiness frameworks at the SME-level are uniform around the globe. The case projects studied in this thesis have this issue – SME eBusiness enablement – as their focus.

Literature shows that in addition to specific drivers and barriers, eBusiness adoption may also be studied in the context of how innovations spread; the diffusion of innovations theory (Rogers 1995) can help in understanding and predicting the adoption of a specific eBusiness technology or application in a specific context, which comprises social systems (adopters) and communication channels (exchange of ideas.)

3. STANDARDS AND STANDARDIZATION

A standard is defined by Feng (2003) as the result of a process in which the form or function of a particular artifact or technique becomes specific. This process by which standards are created is called standardisation. It is not always clear, however, when a standard is indeed a standard and not a norm or habit. Definitions vary, and the very idea of a standard has changed over time. Therefore, in the context of this work, standards are understood as explicit definitions, imposed by a body of recognised authority, and words such as method, practise, etc. are used for other, standard-like, expedients. This also follows a common definition in the literature. (Ensio et al. 2005)

A more practical definition for a standard is, at least in a modern context, that it is a document, created by a body of authority, that imposes rules and features for products, services or processes. In other words, standards yield specifications.

3.1 Features of standardization

The history of standards lies mostly in the field of manufacturing and transportation, where the first widely-adopted standards were developed and deployed. These early standardisation procedures proved quickly that there exists a number of beneficial features that standardization helps to achieve. Research has since established many such properties and standardization is now a well-understood discipline. In order to explain these motives in modern standardization, as well as the functions that standards serve, Feng (2003) has defined in his much-cited study five key features of standardization. These include uniformity, compatibility, objectivity, justice, and hegemony.

3.1.1 Standards as uniformity

Since the very first standards were developed, they have always acted as means for ensuring uniformity in production. Before the advent of the industrialized era, such uniformity was not sought for nor was it generally needed, as early craft industries did not deem uniformity of the products they made necessary; rather, the exact opposite

may well have been true. However, with the production scale-ups that were brought about by the industrial revolution, culminating in the advent of mass production, the need to ensure uniform production at different times and manufacturing sites became paramount.

Much of this need for uniformity has also been highlighted by the arrival of new, disruptive technologies, such as the railroad, the telegraph, and later, information technology. Both the new technologies and the new ideas of how to make business on them have laid the ground for developing new standards. Uniformity is about increased scale, improved predictability – and what is very important – about quality control. The main motive for these features is increased profit. (Feng 2003)

3.1.2 Standards as compatibility

Where uniformity has lain at the heart of even the earliest standardization attempts, compatibility between products and technologies has become increasingly more important with newer technologies, of which the prime example is information technology, where compatibility has become indispensable with the developments in telecommunications, which would not have been possible without this aspect being manifested in ICT standards.

The idea of *network effect* (Katz & Shapiro 1985) says that the value of a network increases with the number of components and users that can work together. This motif can be readily observed in the history and development of the internet, which made it axiomatic that one is able to connect to a system (f.e. a network), and be unrestricted by such factors as the brand of the technology in use. Traditionally, this has not been self-evident: In the past, large telecommunications systems, or large systems in general, had not spanned to such extends, and the users and the products involved were rather homogeneous. Large-scale standardization focusing on compatibility was not needed.

Today, an application of eBusiness, for example, that builds on a plethora of telecommunication and software technologies, requires one to take the compatibility

paradigm onto an even higher level. This development only continues. (Hanseth & Monteiro 1998; Feng 2003)

3.1.3 Standards as objectivity

Above, compatibility has been considered in terms of technology – that two products, for example, may interact by the rules set by standards. Feng (2003) establishes that the more general idea of compatibility can, in addition, be applied not only to technologies but also to facts and claims of knowledge that reach across different locations in the same way computer networks do.

This means that standards and objectivity become one in the sense that if objectivity is understood as the opposite of subjectivity – that the objective claim, unlike the subjective, is independent of any particular individual's opinions – then there emerges a link between standardization and the strive for objectivity, a goal manifested in science, for example, in agreed-on units of measurement, or principles of proof, that allow the comparison of data or results produced at different research facilities. Standardization can be seen as the means for achieving this objectivity. (Feng 2003)

This aspect of objectivity highlights the general trend in standardization now present. It could be argued that the role of uniformity has decreased with the advent of post-industrial, information- and ICT-driven society, where individualization and tailoring, as well as quick-paced manufacturing and production with short reaction times, have become ever more important. On the other hand, this means features enabled by objectivity and compatibility are more and more needed. It is impossible to conceive standards for testing software, for example, without these attributes. Therefore, the move from simple mass production to the state of ever-present flexibility in manufacturing and services is underlined by the quickly risen role of this kind of standardization. (Vuorensyrjä & Savolainen 2001)

3.1.4 Standards as tools for justice or hegemony

The last viewpoint to standardization Feng (2003) offers is that of justice, linked with the idea of objectivity. He argues that although some standardization efforts resulted in colossal, hegemony-like systems where the advertised idea of having everyone stand on an equal footing (nation-wide telephone systems, f.e.) became rather tools for exerting economical power, there has also abounded a sentiment of justice in setting standards.

The roots for an ideological trait in how people view standards are found in the attempts at standardizing units of measurements in France in the 18th century. This was motivated to a great extend by the injustice expressed by peasants who felt the constantly changing (getting larger or measured as most fitting per occasion) units used for collecting taxes or selling goods had become unjust.

Feng (2003) notes that since then, through the history of standardization in the West, similar notions have been attached to standardization. For people to whom "equality" (in which ever context) has meant progress, standards that make things uniform have provided a tool for manifesting these ideas. Of this Feng provides the standardized test in American schools as a typical example.

3.2 Development of standards

Standards are binding. This means that when parties adhere to a standard, they will have to follow that particular specification precisely, to the extend that has been agreed on or the extend to which the specification requires. When standards, having these qualities, dictate issues related to technology, processes, issues on safety, etc., that are fundamental to how companies make business, the general issue of who has the right to enact such vital rules emerges.

The solution lies in part in the definition of a standard: That it is imposed by a body of authority. Most standards in use today are defined by standardization bodies, organizations who are specialised in standardization in general, or within a particular field, and who are recognised by industry or government(s) to have the necessary

expertise and authority to carry out standardization. (O'Donnell et al. 1996; Ensio et al. 2005)

Not all standards are created this way. SFS (2006) accepts as standards also what are called *de facto* standards. These have not been defined by a known standardization organisation, but have been fielded so widely or in so specific a manner in industry that they have in effect become accepted as standards. A de facto standard may also be called an industry standard, especially when there has been an organised effort in creating it instead of its simply becoming one. Gallagher (2007), for example, draws a distinction between a "dominant design" – one that has simply become standard-like – and a true industry standard – one that has been developed formally.

It follows, then, that standardization is not restricted to specific standardization organizations. The role of standards developed outside these organizations is increasing, in part due to the very rapid development cycles in technology industry, where for this reason also the need for standardization has become vast. On the other hand, the type of the standards developed within industry often differs from those created by national or international standardization bodies whose work is much more general in nature and aimed at servicing the needs of society or industry as a whole. (Frank 2002; SFS 2006)

3.2.1 National and regional standardization bodies

Today, almost every country has a single, recognised standards body, whose responsibility it has historically been to develop, issue, revise and interpret standards and standardization work for the needs and special requirements of the given nation. Their role remains much unchanged today, even though the scale and importance of international standardization has increased substantially.

Earlier, these organisations mainly served the need to create standards for use by local industries. Later, their work has included general standards research, interpretation of international standards (official translations, adaptations, etc.), and other similar work. National standards bodies are typically members of ISO (International Standards

Organization) and thereby act as the country's official representative in the standardization conducted by ISO.

Large economies, such as the United States and Japan, typically have a number of such organizations in place of a dominant one. Also, the roles of these bodies and their being in public or private sectors varies. Additionally, in larger economies there also operate separate standards development organizations who cannot set standards themselves but who will aid the national bodies in the actual standards development.

Regional standards bodies share many goals with the national ones, but their reach is that of a specific economical or geographical region, and they may or may not be able to force a standard. CEN (The European Committee for Standardization) is a typical example of such an organisation in that it is private but non-profit, and co-operates and shares work with other regional bodies inside the EU in creating standards that bind all its members with the aim at harmonising the field over the older heterogeneous national standardization. (DIN 2000; SFS 2006; CEN 2009)

3.2.2 International standardization bodies

Where the standards developed and issues by national standards bodies are aimed and tailored for local use, the standards created by international standardization organizations are made available worldwide, free for anyone to use, and so that the process by which they are developed is fully open. Today, many national standards are actually international standards, either adopted directly, or first tailored for local use by the national standards bodies. On the other hand, international standards are on occasion similarly adapted from existing, well-devised national standards, or even industry standards. (DIN 2000; Ensio et al. 2005; SFS 2006)

Amongst the oldest and largest international standards consortiums are the International Organization for Standardization (ISO), the International Electrotechnical Commission (IEC) and the International Telecommunication Union (ITU), all having operated for more than 50 years, based in Geneva, Switzerland. All three have a recognised,

independent status globally. They comprise of various national bodies and smaller specialised organizations.

The standards reviewed in this work are all developed by internationally-recognised bodies. Those referenced along with ISO include IEEE (chapter 5), and OASIS, OMG, RosettaNet, and W3C (chapter 6.) IEEE is a large non-profit organisation that promotes advances in technology through research and by publishing journals and documents on various fields of engineering. IEEE also sponsors conferences and carries out standardization work, in which it has a global status. IEEE now prefers the four-letter acronym as its name so as to not limit itself to its original scope from which it has expanded considerably. IEEE standards on quality management in software development are studied in chapter 5.

OASIS (Organization for the Advancement of Structured Information Standards) is also an international non-profit consortium. Its scope lies mainly in web technologies and eBusiness, subject to study in chapter 6. OMG (Object Management Group) is an industry standards body active much in the same field as OASIS. W3C (World Wide Web Consortium) too is an industry consortium; officially, it publishes recommendations, not standards, but in practice the recommendations often gain an industry-standard-like status and may become part of or form the basis for other standards. W3C is active in internet and web service technologies. RosettaNet and other eBusiness framework providers are considered in more detail in chapter 6.

3.2.3 Standards development process

The procedure by which a standard is created in the large, international or regional standards organizations varies to a certain extent from one organization to another. There is, however, a general model that has been established in the field, and most bodies seem to follow a more or less similar approach in diving the standard-development procedure into chronological phases involving rounds of evaluation between stages from the first drafts to a published standard. (SFS 2006)

In order to provide an overview of a typical international standards-development process, ISO is used as an example. According to ISO, the catalyst to begin the development process comes in the form of a clearly established need, expressed to ISO by stakeholders and sectors. The national member who receives the requirement forwards it to the appropriate technical committee specialising in the area most fitting to the proposed standard. For the proposal to proceed, it has to receive a majority support in the committee; its global relevance and applicability, for example, are evaluation metrics pondered.

Once the work item is accepted, the appropriate technical committee or subcommittee takes over its development. The members of the committee include experts from the industrial, technical and business sectors that have asked for the particular standard. The development continues such that delegations, comprising committee members from the national bodies, meet to discuss and debate until a draft proposal is agreed on. Other organizations may partake, too, and in some cases consensus is already achieved upon its arrival in the committee, based on earlier work, and the proposal may be handed over to a "fast-track" processing. In all cases, the finished draft is circulated as Draft International Standard (DIS) to all ISO members for voting and commentary. If the voting is in favour, the DIS, with possible modifications, is given a Final Draft International Standard (FDIS) status, and circulated for another round of evaluation. Should the FDIS again receive a majority support in voting, it is published as an ISO standard. (ISO 2009)

The above-kind process, involving majority voting, evaluation rounds, and national bodies and committees, seems, and, according to sources close to industry (e.g. Frank 2002; Gallagher 2007) often is slow and heavy from fast-paced technology industry point of view. According to Reid (2008), a typical ISO standard takes over seven years to develop, and an IEEE standard between two to four years with the cost from 2000 to 10.000 USD per page. Still, the need for international standards is recognised by industry. (DIN 2000) This is in part the reason why industry standards and international standards are both being developed, and so that the work may overlap: Industry

standards are available sooner, whereas international standards are more universal and freer of influence exerted by individual parties.

3.3 Summary

A standard is a document, created by a recognised body of authority, that imposes rules and features for products, services or processes. That is, standards yield specifications which all parties following that standard will have to adhere to, either as dictated in the standard or as agreed on by the parties involved. Standardisation bodies can be national or international. Some of the international standards consortiums include ISO, IEC, ITU, IEEE, and many others. In the information technology field there are specialised industry standards bodies such as OASIS or OMG who are much like international bodies but serve the needs posed by their member industries.

Standards can be divided roughly into international, national and industry standards. International standards are free for anyone to use and their development is open. Industry standards, also called de facto standards, may have been developed for such needs of the industry as have not yet been met by an international standard (whose development is slow,) or they may simply have become standard-like due to their wide adoption and success in industry.

Standards possess five key features that explain why they are being created. Standards can be used to ensure uniformity in production – that there are no deviations in the characteristics or quality of products. Standards also provide compatibility between different products that need to be able to work together. The idea of compatibility may also be applied to facts and claims of knowledge, to achieve objectivity in specifications. Finally, standards may become devices for establishing equality between entities as well as become tools for exerting hegemony-like power over others.

In this thesis, the features of compatibility and uniformity are readily observable in standards for eBusiness, studied in chapter 6. The idea of objectivity is evident in standards for testing and quality assurance, subject of chapter 5. The standards in chapter 6 allow the case projects to pursue the concept of equality in eBusiness adoption – a feature found in standards developed for web services. All standards referenced in this study are international and open.

4. TESTING AND QUALITY IN SOFTWARE DEVELOPMENT

This chapter introduces the concepts of software – its design and development – and quality – its meaning and role for stakeholders – and combines the two to provide an overview of quality assurance and testing in software development.

4.1 Software development

4.1.1 Software systems

A software product is a complex socio-technical system, consisting not only of software code but also of hardware and people (stakeholders) in various roles that allow the system to achieve its objectives. (Sommerville 2007) This means that the user is as much part of the systems as is software code or, for example, a network on which the software operates. The systems possesses what Sommerville calls emergent properties, attributes that come into existence not in parts of the system or in its subcomponents, but in the system as a whole as the sum of an integration of all its components and the resultant interaction. Therefore, the final system properties cannot be derived from assessing the parts alone (f.e. code, user, architecture...) This is why many issues in software development are so challenging.

The process of creating a software system is called software engineering or software development. Haikala & Märijärvi (2004) provide a broad-level view on the engineering of a software system as perceived by the developing organisation, shown in figure 3 on page 29. The product development is government by a quality management system that caters to the company management and business goals. For a given software product there are projects, each under its own management, and typically comprising at least the listed general five development stages – also depicted in figure 2 on page 29 – to which affect in various levels depending on the development approach certain immanent activities, such as the project's management, quality assurance, documentation or requirement analysis.

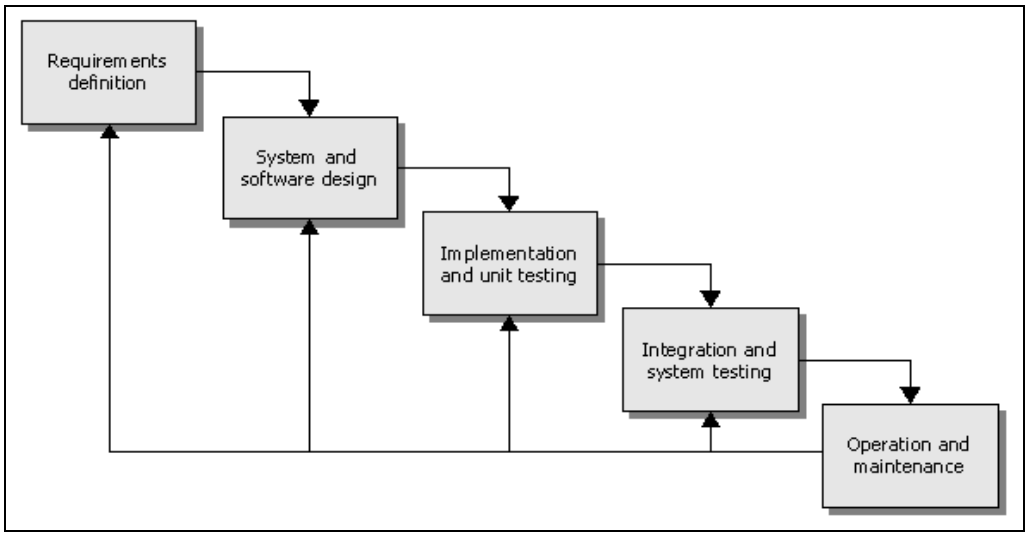

Figure 2: The software life cycle. (Sommerville 2007)

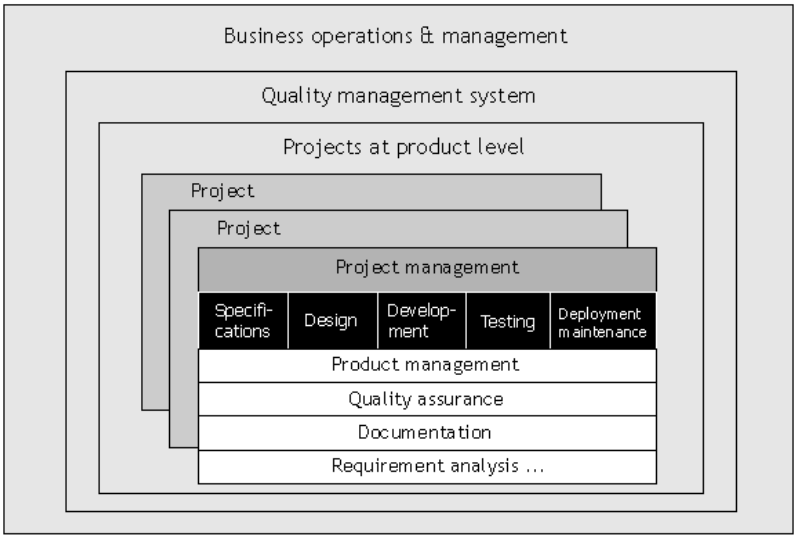

Figure 3: High-level view on software development. (Haikala & Märijärvi 2004)

The software system has a life cycle (figure 2.) It denotes the time span from the point the development of the software begins till the point it is permanently removed from use. As can be observed in the figure, the system need not remain fixed to a certain form while in use, but may, and often do, go through further changes throughout its life. How

much and what kind of changes occur, is related to the objectives of the system as well as its architecture and development process. (Haikala & Märijärvi 2004; Sommerville 2007)

4.1.2 Software development processes

There are many ways in which large software systems can be developed to meet the particular requirements posed by the customer or the perceived markets. These different approaches are called software development processes, and they are often described by models that present them in a much simplified form. Acording to Sommerville (2007), these models provide not only an abstraction (simplification) of the process for easier study and analysis, but also a perspective through which the abstraction is formed. As such, process models are never definite guides for building software but rather general descriptions of the various entities and activities involved.

The waterfall model is the oldest general software development process. It prescribes software to be built in a succession of clearly-defined steps so that each step is finished before the next development phase can be started. It progresses linearly and into one direction like water flowing down a hill. The steps themselves follow the traditional division of software development tasks, shown in figure 2. (Datta 2006; Sommerville 2007)

More modern process models include various modernizations of the classic waterfall method – The Unified Process, spiral model, V-model, for example – and what are called agile methods, of which extreme programming (XP) is perhaps now best known and most practised. Where modern "traditional" models have become more incremental and the waterfall-steps have been replaced with more cyclic approaches with added focus on prototyping, agile methods take this trend even further and have the developers work in quick-paced, small cycles where reaction times to any plausible changes are very fast. (Haikala & Märijärvi 2004; Balijepally & Nerur 2007)

4.2 Quality and software development

The terms quality and quality assurance (including quality control) are essential to software testing and are considered before discussing testing itself.

4.2.1 Quality as a concept

The quality of a software product measures its ability to meet its user's reasonable expectations (Haikala & Märijärvi, 2004.) That is, quality in software development context is not so much about absolute soundness or "goodness" of a specific product, but rather about different qualities that can be measured and evaluated against the user expectations. Quality is therefore not only subjective, but also dependent on the stakeholder – the end-user, for example – and the evaluation environment – what the user expects in the given conditions.

Budgen (2003) and Sommerville (2007) similarly conclude that quality is in essence about meeting customer demands; they add that it would be enticing to quantify quality metrics, such as reliability, security, etc., in a standardised way, but due to quality's being in these areas so abstract a notion, this is very difficult to do. Kit (1995) also emphasises the stakeholders' role in quality, which is, in the end, about customer satisfaction, not about absolute quality metrics. It follows that how quality is assessed will have to be based on customer satisfaction. Kit argues that this creates the framework for quality, and for what is "good" or "satisfactory" quality. Absolute quality is uninteresting.

4.2.2 Quality management

According to Sommerville (2007), quality management is motivated by the idea that although quantifying quality metrics is difficult, it is possible to achieve (or approach) higher quality in software development by placing standards and organisational quality procedures that encapsulate sound practices for given tasks and which, when adhered to, lead to higher quality products. Sommerville adds, though, that so simple a view is not

shared by all, and that there are attributes in quality management that cannot be standardised or subordinated to procedures.

The objective of a quality management system is to ensure that the software development process achieves the desired quality level within the specified time and budget constraints. This quality level includes, depending on the definition, product-specific customer demands and / or the quality of the development process itself. (Sommerville 2007; Jäntti 2008)

A typical, modern quality management system leaves room for team managers and experts to tailor their methods as most applicable to a given task. The systems do, however, take more strict measures on providing demonstrability – to be able to prove that set standards or practises have been followed. (Haikala & Märijärvi 2004)

Quality management proceeds parallel to the development process; this is depicted in figure 4. Quality assurance and control, and quality planning, also shown in the figure, are part of the quality management process.

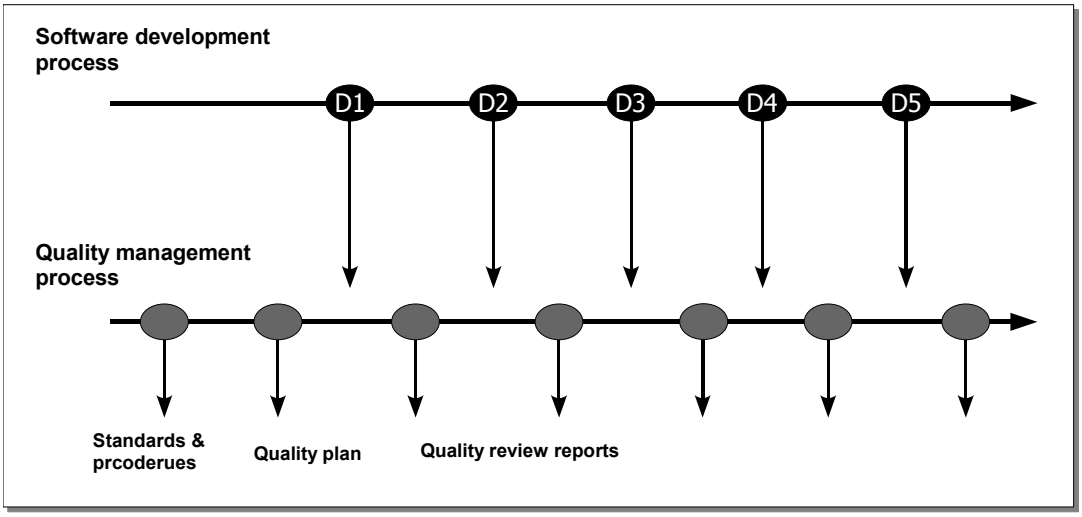

Figure 4: Quality management and software development. (Sommerville 2007)

4.2.3 Quality assurance and control

Quality assurance (QA) encompasses the specific activities that enable the quality management process to achieve the targeted quality. Quality control, respectively, determines the results of the work products to ensure that they conform to the standards or requirements specified by the quality management system. (Perry, 1995)

Haikala & Märijärvi (2004) agree with this definition but point out that the terminology and exact definitions used on discussing QA vary. For some, quality assurance is synonymous with the general concept of quality management; for others, QA is the activities that together with quality control make the quality management process, as shown in figure 4 and similarly defined by Perry (1995) and Sommerville (2007). Jäntti (2008) further divides the terminology into a more traditional, software-building-centric QA and into services-centric QA that is suited for maintenance and tasks external to most of the traditional development process stages. The discussion in literature is, hence, much dependent on the particular view adopted.

The definition followed in this work is derived from Kit (1995), Taipale (2007) and Bertolino (2007): Quality assurance and quality control are quality management processes that include testing as a tool. This means that all testing is conducted as dictated by the organisation's QA practises and policies, but not all QA need to be testing.

4.3 Software testing

Testing is, according to Mayers et al. (2004), “the process of executing a program with the intent of finding errors.” The definition implies that testing is not about showing that there are no (longer) errors, or that something “works;” the focus is on detecting mistakes. The authors believe this is very important because human work is goal-oriented. If the purpose were to show there are no errors, exactly that would likely results, whether there be errors or not. The Mayers et al. (2004) definition is called negative testing (Hämäläinen, 2005), and it includes finding mistakes also external to requirements and specifications.

Kit (1995) has pondered a number of other definitions found in the literature. He concludes that from the tester's point of view, the best definition is still the negative one, but with emphasis that testing is the process to discover every conceivable fault or mistake in a work product.

4.3.1 Errors and risks

Software is written by people, and people make mistakes. This is included in the definition for testing, and testing cannot prevent these errors from occurring. Testing can, however, locate errors soon after their introduction to the work product and thus prevent their effect from cumulating. The cost of errors that will have to be fixed at a later stage than they were introduced grows rapidly as the error accumulates into work products that built upon the erroneous one. Errors also deteriorate quality, and this effect is similarly cumulative. Hence, finding errors early and in an efficient way is the most important objective for testing. It follows that successful testing is imperative for successful quality management. (Kit 1995; Sommerville 2007)

In practice, the purpose of testing cannot be to discover every error introduced to the product, as this would not only be too expensive, but also impossible in a large software system. So, decisions have to be made as to what is tested, and how throughly, based on the resources available.

When risk is used as the basis for making these decisions, they become more rational, and testing will focus on those parts that are most used, or which have the most severe consequences in case they should fail. How and what kind of risks are evaluated, is product and customer dependent. In addition to risk-analysis, it is also good to focus on parts that simply are likely to have errors in them. (Kit 1995; Perry 1995)

4.3.2 Verification and validation

Verification and validation (V&V) is the name given to the processes by which it is evaluated if the software product being developed is conforming to its specification and delivering the functionality requested by the customer. (IEEE Std 1012) The standard

definition does not restrict V&V to testing. However, according to Kit (1995) and Taipale (2007), all testing is verification or validation. This view is adopted for this thesis, and it tallies with the definition in chapter 4.2.3. Therefore, all testing activities are either of verification or validation type, and all test objects (software code, documents, etc.) can be subject to V&V.

Verification is called by some people “human testing,” as in contrast to validation, it typically involves studying documentation. Verification is a process of evaluating, inspecting, reviewing, and doing desk checks of work products such as requirements and design specifications, and software code. For code it means a static analysis, called a code review, not dynamic execution of the code. Verification ensures that the output of the stage is according to what is expected. It tells if the product is being built right. (Kit 1995)

Validation typically involves running software code or a simulated mock-up. It is therefore dynamic and more automated, computer-centric. Validation often reveals symptoms caused by errors, and helps to understand whether the right product is being built. For effective testing, both V&V activities are required, and they complement one another, making the other more effective. (Kit 1995)

4.3.3 Testing methods

The high-level division of testing methods is into black-box and white-box testing. Black-box testing treats the program as a single “black box” whose behaviour and structure are unheeded as if they were unknown to the tester. As the box is fed with test inputs, derived from the software specifications, its output is observed in order to find any deviations from the specification. In case this approach were used for finding all errors in the software, exhaustive search could be conducted by using all possible inputs. This is, however, often infeasible.

White-box testing permits the tester to study the program's internals during the test. The test data are derived by examining the program's logic, supported by its specification, and the test cases that result may be written out as flowcharts that depict the path under

test. As with black-box testing, the test cases have to be hand-picked, as exhaustive testing of all possible cases is no more feasible than it is with black-box testing: The number of possible execution paths inside the software grows exponentially with the software size. (Bazer 1990; Mayers et al. 2004)

Verification and validation testing methods include different kinds of activity that form the phases in the testing life cycle, as shown in figure 5 on page 37. The testing life cycle will adhere to the software life cycle such that for each phase of development there is a corresponding phase for testing. (Kit 1995)

The most common methods for verification include technical reviews, walkthroughs, and inspections. Inspections are, according to Kit (1995), the most formal and structured verification method. The objective of an inspection is to discover and discuss defects in the product under review and also to share information between people and collect data. This often happens in the form of a meeting for which the participants prepare according to their role in the review.

Figure 5 shows how validation tests are normally phased differently in the life cycle to verification. This has to do with the objectives of validation: It checks that the requirements are met. This entails that at least two general test types are developed: Tests that will determine if the users' requirements are satisfied; this is based on the requirements specification. And tests that will determine if the product's actual behaviour is according to the functional design specification. (Kit 1995)

Irrespective of the life cycle model the testing will assume, a number of general stages are normally found within the testing process (figure 5.) The most used validation test are listed below.

System testing assesses the results of system building, which connects related components so that the system as a whole will work as expected. System testing is further divided into integration testing, which tests multiple components together and tries to find defects in their interaction, and to release testing, which tries to find defects in a software version that could already be released to users. System testing has grown

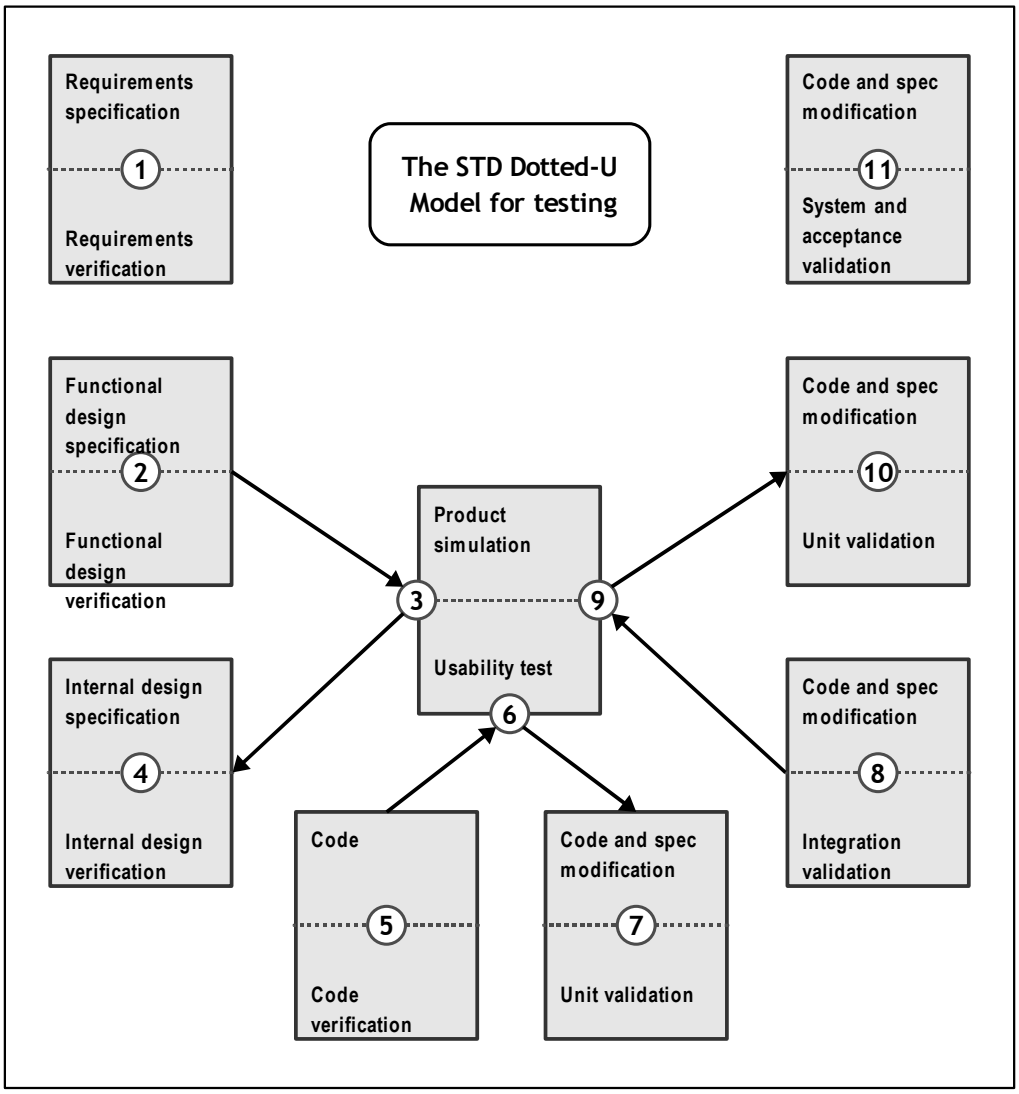

Figure 5: The Software Development Technologies Dotted-U Model for testing. (Kit 1995)

in importance, as today more and more systems are attempted to be built on the re-use of earlier modules and components. (Haikala & Märijärvi 2004; Sommerville 2007)

In component testing, a single module is tested with no heed to other units; this is also known as unit testing. Efficient component re-use will reduce unit testing or make it altogether unnecessary, shifting the testing efforts towards integration and other kinds of testing. (Sommerville 2007) Performance and reliability testing are concerned with the emergent properties discussed in chapter 4.1.1.

When errors are fixed, the changes may necessitate other changes elsewhere, or incur new errors in related parts. This entails that other components or the whole system will have to be tested as well. This is known as regression testing, and unless it can be automated, it can be very expensive. (Haikala & Märijärvi 2004)

4.3.4 Testing tools and measurements

Testing tools make testing easier and more efficient (Kit 1995.) The use of testing tools has increased as the tools have improved, and on the other hand, quality requirements have increased. New tools have also made new kinds of testing, such as automated testing or various types of regression testing, possible. (Ernst & Saff 2004)

The most traditional testing tools are familiar from software development: Editors, debuggers, schedulers, static and dynamic code analysers, and similar basic tools do not require special testing know-how and are widely adopted. More advanced versions of these tools include syntactic and semantic analysers that may work based on the requirements specification or the code alone. They try to find problems or errors beyond the capabilities of normal compilers. (Pesonen 2003)

Tools for test planning help in defining the scheduling, the scope, and the approaches as well as the resources needed for testing activities. Test design tools are much like planning tools, and as Kit (1995) notes, can typically offer only limited help in these very mentally-oriented activities. The tools help somewhat by providing test data and coverage analysis and similar features. Many of these tools may help also in running the actual tests, and then, by the help of data analysis, make evaluating the results easier.

There are still more tools that may be useful for the testing effort. These include software that help managing defects via bug tracking systems and software that helps in version control and configuration management. The most modern testing approaches – automated, continuous testing, and automated regression testing – require their own, sophisticated tools. (Kit 1995; Liu 2000; Pesonen 2003)

Kit (1995) notes that although measurements provide answers in testing, everything cannot be measured as there are too much data. He suggests a number of useful measurements, including software complexity measures, verification efficiency measurements, tracking bug reports, and measuring test coverage together with the execution of test cases. More elaborate rationale for each measurement is provided by Kit (1995). By collecting the right measurements data, the tester will be able to better predict the progress in the development cycle and assess the efficiency of the current testing plan. Therefore, measurements are vital for efficiently managing testing. This way, they also help in managing the software development process as a whole. (Pesonen 2003)

4.4 Summary

Software products are complex socio-technical systems composed of software code, people, hardware, and many other factors, all of which together allow the software product to achieve its objectives. Because there are so many interrelated attributes that affect the software system, various development processes have been devised for managing the development effort. This way, the software assumes a life cycle its development will adhere to.

The purpose of software development is to meet user expectations; this is measured by quality. Hence, high quality is achieved by fulfilling customer requirements, not by building "good" quality as measured in absolute terms. In software development, quality management is a process by which it is ensured that the targeted quality is achieved. Quality assurance and control are quality management procedures. Quality assurance encompasses the specific activities that enable quality management to achieve the targeted quality, whereas quality control ensures that the work products conform to these requirements. Both make use of testing, but are not limited to it.

The purpose of testing is to find errors in a work product. Since it is impossible to locate all mistakes and faults in large software systems, risk-analysis is used to decide how resources are spent on testing different parts or aspects of the software. Testing is divided into verification and validation (V&V.) Verification tells the tester if the product is being built right; validation tells if the right product is being built. V&V can further be divided into specific testing methods. Testing is governed by a testing life cycle which is based on the software life cycle stages. Testing tools help in managing and conducting testing and make certain new kinds of testing possible. Measurements too aid in test management by collecting data on testing activities. By them, testing also benefits the software development process as a whole, making it more predictable.

Standards for testing and quality assurance and control in the software development context are studied in chapter 5.

5. STANDARDS ON QUALITY MANAGEMENT AND TESTING

This chapter provides a study on open, international standards on managing quality and testing in software projects. The goal was to find what standards are now available in the field, how they approach their objectives, and how these standards relate to one another in the larger scheme of QA. This review supports the conclusions made in chapter 7.

The chapter is divided into two main parts, ISO and IEEE quality management, inside which select standards are reviewed in greater detail together with an overview of each standards family.

5.1 ISO approach to quality management

5.1.1 ISO 9000

The 9000 series of standards is a general description of quality assurance elements that are applicable to all industries irrespective of the products or services they offer. The standard focuses on the quality assurance system and defines the requirements that it must meet. The standard does not, however, specify how these requirements should be met. To receive an ISO 9000 certification, the quality system has to be assessed and approved of by an outside auditor.

ISO 9000 has historically included standards numbered from 9000 to 9004, but 9002 and 9003 have been integrated to 9000 and 9004 and are no longer maintained separately. The series is revised regularly to conform with the developments in the field, and the latest revision was made in 2008. Of the 9000 series, ISO 9001 is today the most often used in software industry. It covers documentation, design, testing, development, production testing, installation, services and a number of other related processes. ISO 9000-3 provides specific guidelines for applying ISO 9001 in the software development industry.

5.1.2 ISO standards family on QA

To supplement the rather general QA guidelines as described in the 9000 series, a number of other ISO standards have been devised to provide more elaborate specifications on managing quality as perceived by the software industry.

The most important of the currently available standards is still ISO 9126, which defines quality metrics and their evaluation in software development. This standard is studied in more detail in chapter 5.2. A major new multi-part standard, ISO 29119, which caters to the most important aspects not yet covered by ISO standards, is now under development. Its current status is reviewed in chapter 5.3. ISO 14598, which specifies evaluation processes for the software product, references and is based on ISO 9126. This relationship is is illustrated in figure 6 on page 43 and explained in chapter 5.3. ISO 15939 specifies measurements for the software development process itself. Parts of this standard have been added or integrated into the latest revisions of other, development process and product life cycle related standards which ISO now has a number of. The latest 9126 DIS that is subject to study in this chapter has also inherited some new aspects from ISO 15939. Work has started on a new standard series ISO 25000, which is to eventually supersede ISO 9126 and 14598, combining them into one.

ISO 8402 has been used to define the vocabulary for standards on quality management. However, as of the latest revision of ISO 9000, 8402 has been superseded by ISO 9001 and is no longer maintained. A number of standards on quality management systems themselves exists, too – mostly found in the 10000 series – and there is also an ISO standard for QA system auditing, ISO 19011.

Finally, ISO 15504, known as SPICE, provides a standard for evaluating processes, based on earlier ISO life cycle standards.

In chapter 5.5, select IEEE/ANSI standards are compared to the above-listed, corresponding ISO/IEC standards. Next, the standards ISO 9126 and 29119 are studied in more detail.

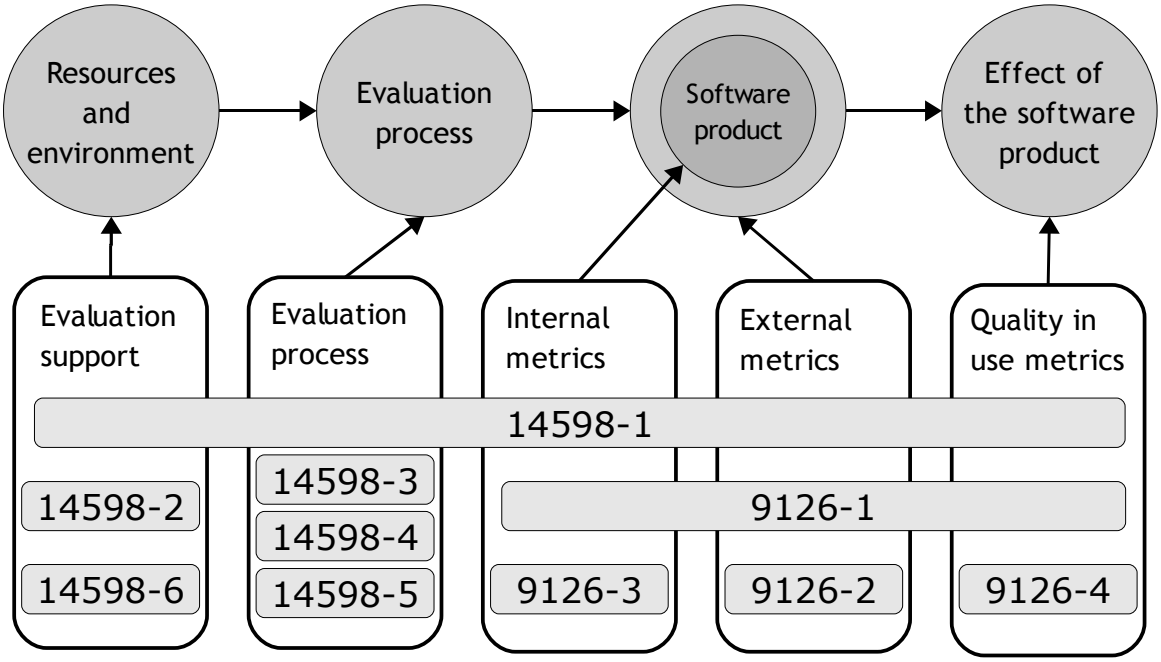

Figure 6: The relationship between ISO/IEC 9126 and 14598 standards. (ISO FDIS 9126-1 2008)

5.2 ISO 9126

ISO standards that build on the 9000 series divide quality in software development into two categories: Product and process quality. ISO 9126 is concerned with product quality while process quality is treated in ISO 15504. Product quality is further divided into an evaluation process for software products, ISO 14598, and quality in software products, ISO 9126. The standards are closely related (figure 6.) It can be observed that the focus of ISO 9126 lies in defining the quality model and its metrics, whereas 14598 builds on this specification and defines the process for evaluating quality. Figure 7 on page 44 further describes this relationship by illustrating how an evaluation process, defined by 14598, may use the 9126 model. Similarly, many kinds of evaluation process could be derived from 9126, and some have been proposed in the literature.

ISO 9126 consists of four parts: Quality model, external metrics, internal metrics, and quality in use metrics. These are treated in more detail below.

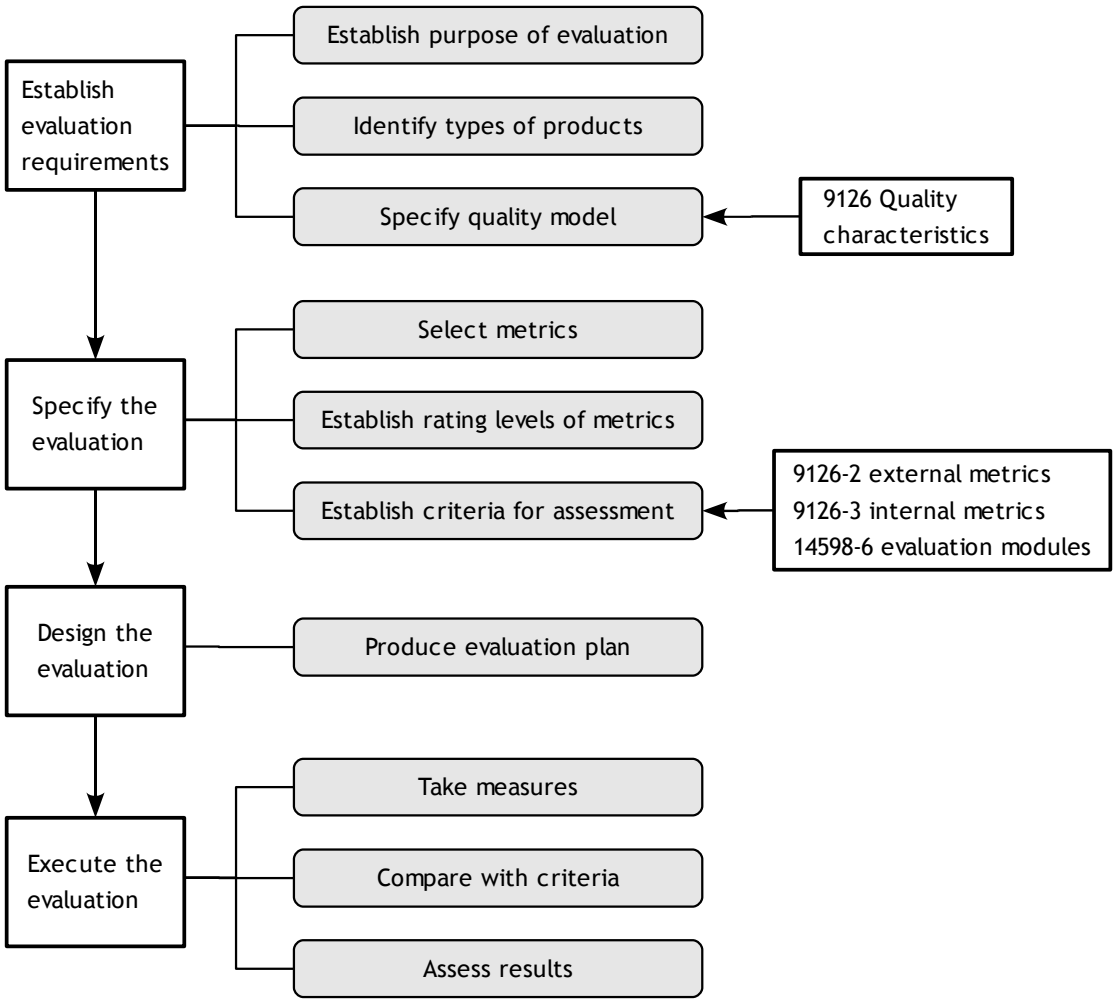

Figure 7: ISO 9126 quality model for ISO 14598 evaluation process. (Gruber et al. 2007)

5.2.1 Quality model

The first part establishes a model according to which the later parts specify quality metrics. This model is applicable to all other ISO standards that may have to reference one. The model consists of six quality characteristics, each further divided into sub-characteristics, categorised into a table on page 45. The sub-characteristics are still further divided into attributes, but the standard does not define these, as they are implementation specific. Attributes are elements in the software that may be measured or verified. Attributes are also used for determining the software's capability for a given

Table 1: Software quality characteristics according to the quality model defined in ISO 9162-1 DIS 2008.

Characteristic	Sub-characteristics	Description
Funtionality	Suitability	The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.
	Accuracy	
	Interoperability	
	Compliance	
	Security	
Reliability	Maturity	The capability of the software product to maintain a specified level of performance when used under specified conditions.
	Recoverability	
	Fault Tolerance	
Usability	Understandability	The capability of the software product to be understood, learned, used and be attractive to the user, when used under specified conditions.
	Learnability	
	operability	
	Attractiveness	
	Usability compliance	
Efficiency	Time Behaviour	The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions.
	Resource Utilisation	
	Efficiency compliance	
Maintainability	Analysability	The capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.
	Changeability	
	Stability	
	Testability	
	Maintainability compliance	
Portability	Adaptability	The capability of the software product to be transferred from one environment to another.
	Installability	
	Co-existence	
	Replaceability	
	Portability compliance	

Table 2: Quality in use sub-model characteristics as defined in ISO 9162-1 DIS 2008.

Characteristic	Description
Effectiveness	The capability of the software product to enable users to achieve specified goals with accuracy and completeness in a specified context of use.
Productivity	The capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.
Safety	The capability of the software product to achieve acceptable levels of risk or harm to people, business, software, property or the environment in a specified context of use.
Satisfaction	The capability of the software product to satisfy users in a specified context of use.

quality (sub-) characteristic. Chapter 5.2.2 gives more details on how the internal and external metrics are employed.

Along with the quality model, the standard also defines a “quality in use” sub-model. This is quality as perceived by the user. Quality in use is dependent on external quality, which in turn is dependent on internal quality. This means that in order to achieve the desired quality in use, one will first have to establish the supporting internal and external quality, and in this order. The quality is established by quality measures, explained next in metrics. The characteristics which quality in use model comprises are listed in table 2.

External metrics use measures derived from the behaviour of the software and the system it is part of. This involves testing, operating and observing the executed software or the system. The standard suggests the software be evaluated based on such external metrics as affect business objectives that involve using, exploiting, and managing the product in a specified organisational or technical environment. As with internal metrics, external metrics allow stakeholders to evaluate and establish quality levels during testing or operation. Quality in use metrics are more specific external metrics concerned with select use-scenarios, based on the characteristic listed in table 2.

5.2.2 Metrics

The metrics in ISO 9126 DIS 2008 are either internal or external. Quality in use metrics can be seen as forming a sub-category inside external metrics, and are treated here as part of external metrics. Internal metrics apply to the non-executable part of the software product, such as specification documents or source code. The standard recommends that all intermediate products should always be evaluated by internal metrics; this should also include properties derived from simulated behaviour. Since external quality and quality in use are dependent on internal quality, internal metrics are used to ensure that the required external quality is achievable. Therefore, internal metrics help the user, the developer, the evaluator and the tester to better assess and address quality issues prior to the software's becoming executable.

ISO 9126 has the internal and external metrics built in the following way: Based on the quality requirements devised for the software product, quality characteristics and sub-characteristics are listed according to the standard quality model (table 1.) Then, external metrics are derived and quantified (with specific ranges for values) from the list such that they may be evaluated against the user requirements. Following this, the internal attributes for the software are specified so that the newly-defined external quality and quality in use can be built by them into the product. Finally, appropriate internal metrics are defined and quantified so that the internal attributes can in turn be used for evaluating whether the intermediate software products meet the internal quality specifications.

Once the characteristics are known, their relationship to one another should be representable as a tree which has the characteristics placed at the top and the attributes at the very bottom, with sub-characteristics in between. In the tree, an attribute may be part of more than one branch: An arbitrary number of attributes may affect an arbitrary number of internal and external sub-characteristic. That is, the relationships can be many-to-many.

Although the standard does not discuss V&V specifically in relation to evaluating quality metrics, the analogy to verification and validation as defined in chapter 4 is obvious: Verification is concerned with internal metrics while validation deals with external and quality in use metrics.

A prevailing feature of the 9126 standard is that it has all the quality assessments done strictly on quantified metrics. On the other hand, how to quantify the metrics is left for the implementer to decide, as well as how to assess the enumerated values. The standard does list, however, a number of general requirements for metric-values in case they will be compared between different products. This is to ensure the comparison is feasible.

ISO 9126 parts 2, 3 and 4 define a set of external, internal and quality in use metrics, respectively. This listing is not inclusive but rather suggests a best practice for defining metrics to be used along with the ISO 9126-1 quality model. The standard encourages

the user to modify and supplement the provided metric tables as most fitting per the requirements of a given software product.

5.3 ISO 29119 and BCS 7925

Although ISO 9126 and 14598, especially when coupled with a software life cycles standard such as ISO 12207, form a coherent view to quality, the ISO family is lacking a more definite standard on V&V, i.e. the actual testing techniques; IEEE, for example, has devised standards to answer these needs (Std's 1012, 1028, 1008, in particular), although even these standards only cover a limited scope of software testing. On the other hand, British Computer Society (BCS) has sponsored an effort to develop a standard for component testing and testing terminology, BCS 7925, which was published first time in 1998 after an eight-year-long development phase. Having recognised the need for BCS 7925 kind of testing-activity-specific standards as well as the void now present in ISO standardisation for most aspects on software testing, ISO has started the work on a new standard ISO 29119, with support from IEEE, in order to expand on the work already done on BCS 7925 and to provide, once finished, a coherent international standard for testing techniques, processes and documentation. Due to the significance of this development, BCS 7925, which is already well established, is introduced in this chapter with discussion on the characteristics the latest ISO 29119 draft versions show the standard is expected to assume. (Reid 2008)

5.3.1 BCS 7925

BCS 7925 comprises two parts, documentation and testing techniques. It is fully published and mature, and is adoptable to projects where the larger software testing strategy is already in place and where BCS 7925 can be plugged in for its specific testing tasks.

The part for testing techniques, called component testing, has as its goal not only to define techniques but also to make the use of the standard auditable and the testing itself measurable so as to aid in improving the organisation's testing processes. BCS 7925

concentrates in testing techniques solely on component testing in order to limit its scope to a practical and manageable range. In this regard, it is comparable to IEEE Std. 1008 on unit testing.

Conformance to the standard is determined in the form of a testing process, defined in chapter 2 in the document BCS 7925 Standard for Software Component Testing. The process presupposes that upon starting this process a component test strategy and plan are already specified, details for which are provided in the standard. Based on these, a component test specification is then devised, describing selected test case design techniques and the test cases themselves with such characteristics as the objective or the initial state determined. The standard further prescribes, heeding the need for measurability, that the execution of each test case has to be repeatable and that all test cases must be executed during the process. When a test case is run, it is recorded together with a number of data so that its result can be measured against the test-specification-defined expected result. Included in these data are at least all test coverage criteria for a given test case. For the process to be according to the standard, it will have to continue until all completion criteria for all test cases are met. More details on the specifics are available in the standard.

It is noteworthy that the standard specifies the design techniques amongst which the implementer chooses the technique for each test case. For conformance, no other techniques may be used. This is also true for the test measurement techniques provided. This way, BCS 7925 is more strict than some of other standards studied in this chapter; this is related to the authors' goal of making BCS 7925's readily auditable.

5.3.2 ISO 29119

Figure 8 on page 50 illustrates the four parts ISO 29119 comprises and where each part is derived from. The first part of BCS 7925, vocabulary, forms the basis for the first of the four ISO 29119 parts. The second and third ISO 29119 parts, testing techniques and testing processes, inherit, in turn, the respective aspects of BCS 7925's second part (discussed in 5.3.1.) Finally, the last part, documentation, will be based on IEEE Std. 829 (table 3 on page 52.)

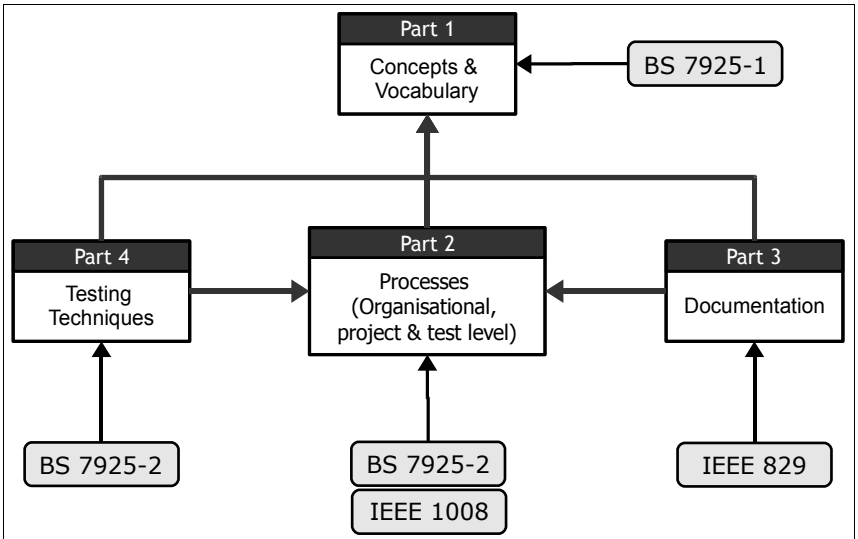

Figure 8: ISO/IEC 29119 overview. (Reid 2008)

As can be seen, ISO 29119 is a large standard. Therefore, in contrast to the standards it builds on, its conformance does not require the user to implement all the standardised processes or use, for example, all the documents specified; rather, the standard allows various levels of conformance: For a *full* conformance, the user declares a set of processes and documents for which they show that the standard-specified requirements have been met, evidenced by the process outcomes. For a *tailored* conformance, the user declares a set of ISO 29119 clauses, some or all which may have been modified, and shows that the documents and the processes are according to this set.

The standard views testing similarly to IEEE (chapter 5.4) in that testing is on a high level verification and validation and that the term "testing" denotes activities and techniques governed by testing processes within either V&V context. The standard-specific concepts are explained in the first part; it also includes general information on testing and provides an overview of how ISO sees testing best managed in the context of software life cycles and project management. Based on the draft version available, it is not yet clear which sections are categorised as informal and there is still some inconsistency in the terminology used.

The second part defines testing processes. The ISO model has all testing governed by a top-level (management-level) testing policy or one or more organisational testing

strategies, to which all project-level testing adheres. This relationship is reflected in the test plan, defined in the third part of the standard. Organisational testing policy is a process by which it is determined what testing is carried out in the organisation. It also establishes testing practises and specifies how this policy can be monitored and improved. Organisational testing strategy then defines how testing is carried out in the organisation. Two other processes, project test management and test levels, are also provided. The aim of project test management is to define a process for the planning, strategising, monitoring, and the control and reporting of testing at the project level. Test level processes define how testing is conducted at a given level (e.g. unit, integration, system, acceptance testing) or for a particular type of testing (e.g. performance, security or functional testing.)

Although no draft was yet available for the fourth part, testing techniques, an overview provided in the first part shows that ISO has included and defined all well-known verification and validation methods, divided into static (walkthroughs, inspections, etc.) and dynamic testing techniques, the latter of which include some not part of BCS 7925.

ISO 29119 expands the standards it is based on by defining all common V&V techniques for all testing levels. It determines how testing is managed in the software life cycle context and provides tools for test planning and documentation. This is a major philosophical change from the way IEEE or ISO have approached standardization for QA. In place of a more or less integrated set of individual standards for specific needs or tasks there is now, for the first time, emerging a standard that collects related aspects under a single, coherent view. In practice, ISO 29119 might, once finished, replace for the tester all of the individual IEEE standards studied in chapter 5.4 together with some listed in table 3 on page 52. (ISO/IEC 29119 Part 1 Draft 2008-16-12; ISO/IEC 29119 Part 2 Working Draft 2008-08-22; ISO/IEC 29119:2010 Part 3 Final)

5.4 IEEE standards on quality management

The IEEE standards family on software quality management comprises a larger number of individual standards than that of ISO. On the other hand, IEEE standards seem to be

focused on individual needs within the larger quality management field; hence, the user is expected to select the most fitting standards for their particular needs. This way, IEEE standards may be plugged in to a mix of standards from various sources. Table 3 collects the most relevant IEEE standards for an easier overview. A more detailed study on select standards follows.

Table 3: IEEE standards family on quality management

Standard	Revision	Description
730	2002	Specifies the minimum requirements for the creation of quality assurance plans.
829	2008	Defines the format for documents used in the eight stages of software testing. The stages are also specified by the standard.
830	1998	Provides the IEEE recommended approach for authoring software requirements specifications. Defines also the model the recommended practice is built on.
1008	1987	Defines an integrated (systematic, documented) process for unit testing with a minimum set of tasks for each activity. Reaffirmed in 2002.
1012	2004	Standard for software verification and validation. Defines the processes for both V&V in terms of activities and related tasks. Defines also the IEEE V&V plan and its format.
1028	2008	Standard for software reviews and audits (inspections.) Defines the five IEEE software audit and review types and the procedures for executing them.
1044	1993	Classification for software anomalies. Std 1028 and 1044 are recommended by IEEE to be used together. Std 1044 supplements Std 610.12
1061	1998	Provides a methodology for establishing quality requirements and identifying, implementing, analysing and validating process and product quality metrics. The metrics themselves are not specified.
1233	1998	Guide for developing system requirements specifications. More general than Std 830. May be combined with Std 830 in the future.
610	1990	610.12 is the IEEE standard glossary of software engineering terminology.
Other standards		Like ISO, IEEE also has standards applicable to software life cycle and the development process management. For example, ISO 12207 life cycle standard is also an IEEE stabdard by the same number. Many such standards are referenced by the ones listed above, but these are not treated in greater detail in this chapter.

Many of the IEEE/ANSI standards (but not necessarily recommended practices or guides) provide the specification in the form of "shall," "should," and "may" clauses. In these cases, a conformance to the standard may be claimed when all requirements, indicated by the "shall" clauses, are met. IEEE standard format conformance is achieved when the results generated by applying the standard are according to the "shall"

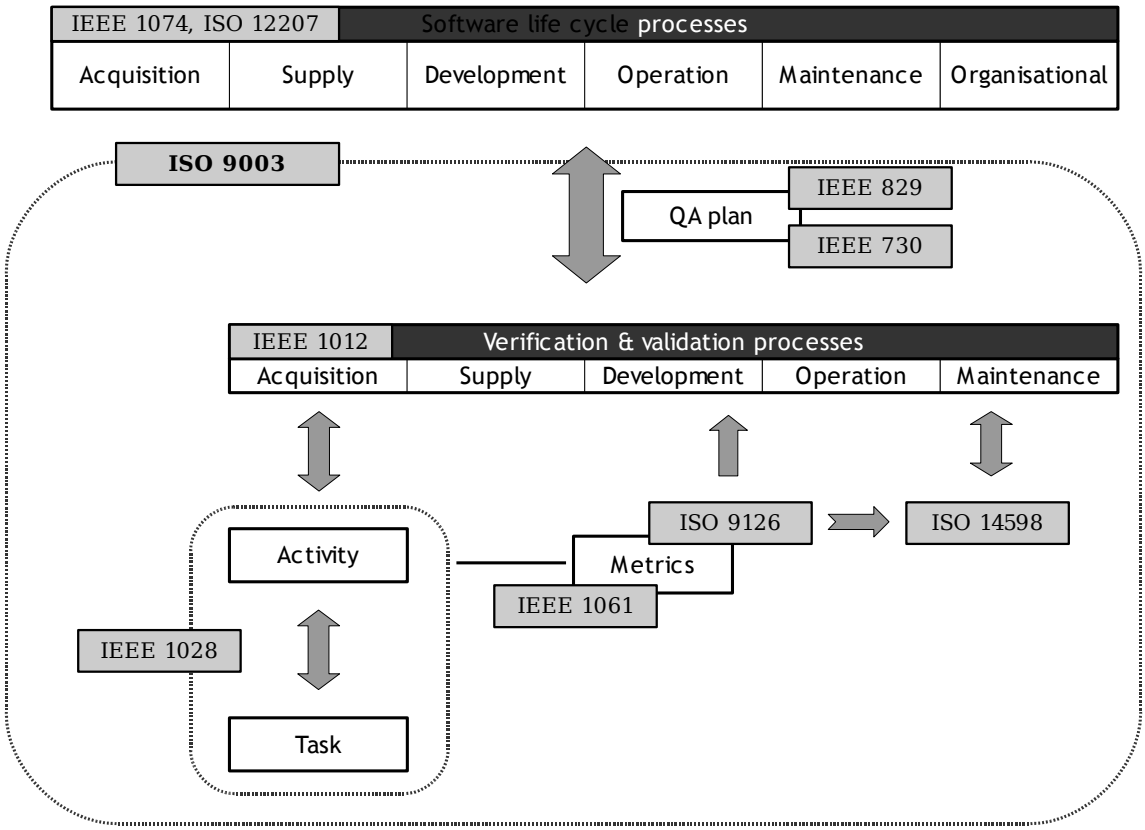

Figure 9: Select IEEE and ISO QA standards mapped.

requirements. “Should” clauses provide recommendations, and “may” clauses express alternative or optional methods for satisfying a requirement. IEEE standards typically also provide a good idea of if, or how, they may be supplemented by other standards or non-standard methodologies. IEEE standards mostly reference other IEEE standards, but occasionally ISO standards are also referred to.

The standards explained in chapters 5.4.1 to 5.4.3 provide the means for evaluating the IEEE approach to quality similarly to ISO 9126 and 14598 now do for ISO. As a result, it becomes easier to compare these two approaches and, with Std 730, see where the other IEEE standards studied here position in the larger IEEE quality management map. Figure 9 shows the standards studied here mapped logically to one another.

5.4.1 IEEE Std 730 Software quality assurance plans

The Std 730-2002 applies to the development of a software quality assurance plan (SQAP.) It does not impose restrictions on the use of other similar standards in a compliant SQAP, nor is its use enforced over another standard. IEEE recommends the user to exercise their own judgement in whether to apply the standard for a given software project. Conformance to Std 730 is according to the principle of requirements and recommendations, given in chapter 5.4. The other standards studied in this chapter will be placed according to the sections provided in the table, making it easier to see their relationship to the IEEE quality management approach.

The content for the SQAP document consists of 16 mandatory sections – listed in table 4 – plus any additional sections, included as necessary. Although it is not possible to omit a mandatory section, it is possible to omit information for a section. In this case, an explanation for this has to be included under the section title. The standard also requires that the SQAP be approved of by each manager of each unit in the organisation having responsibilities defined in the SQAP.

Table 4: Software quality assurance plan as defined by IEEE Std 730-1998

Section	Purpose
Purpose	Defines the purpose and scope of the SQAP; lists the software items and their intended use; for each item, defines the pertinent software life cycle portion.
Reference documents	Provides a complete list of all documents referenced in the SQAP.
Management	Describes the project's organisational structure, as well as pertinent tasks and roles and responsibilities. The adopted viewpoint is quality management centric.
Documentation	Identifies all documents governing development, V&V, use and maintenance. Lists documents that shall be reviewed or audited, and for each document defines the auditions and reviews with their criteria.
Standards, practices, conventions and metrics	Identifies standards, practices, conventions and statistical techniques to be used. Identifies the metrics and quality management to be applied. Product and process measures may be included. States how conformance to these items is to be monitored and assured.
Software reviews	Based on Std 1028. Defines the reviews to be conducted. Provides a schedule and states how the reviews shall be accomplished. States what other actions may be required to support the reviews.
Test	Identifies all tests not included in the software V&V plan but covered by SQAP. Shall define the methods to be used. A possible separate test plan is referenced.

Problem reporting and corrective action	Describes the practices and procedures to be followed for reporting, tracking and resolving problems or issues in both work products and processes. Organisational responsibilities and their implementations are also stated.
Tools, techniques and methodologies	Identifies the software tools, techniques and methods employed in order to support SQA processes. Reports the intended use, prerquisities, and applicability for each.
Media control	Lists the methods and facilities to be used for identifying the media for each work product and document, and the related copy and restore processes. Defines how the physical media are protected from damage and unauthorised access.
Supplier control	Establishes the provisions for ensuring that software provided by a supplier is according to requirements. Establishes also how the supplier will receive adequate requirements. If old software is re-used, its applicability to SQAP is detailed here.
Records collection, maintenance and retention	Identifies the SQAP documentation to be retained. States the methods and facilities to be used to assemble, file, safeguard and maintain this documentation, and shall designate the retention period.
Training	Identifies the training activities necessary to meet the needs of the SQAP.
Risk management	Specifies the methods and procedures employed to identify, assess, monitor and control areas of risk during the SQAP portion of the software life cycle.
Glossary	Explains terms unique to the SQAP.
SQAP change procedure and history	Contains the procedures for modifying the SQAP and maintaining a history of changes. Contains also a history of such modifications.

5.4.2 Standards on verification, validation and inspections

IEEE has devised separate standards for V&V, Std 1012-2004, and inspections, Std 1028-2008. The standards are logically related but not bound by IEEE to one another – both standard may be used together with any other standard(s) the user sees fit.

Std 1012 is a process standard that addresses all software life cycle processes, being compatible with most generally-available life cycle models. The use of ISO/IEEE 12207 is recommended, though. The standard is applicable not only to software being developed, but also for acquisitions and maintenance. The term software encompasses all types of software code and the related documentation. To the extend required here, the way Std 1012 defines verification and validation is according to the definitions provided in chapter 4.

The standard uses software integrity levels to determine the V&V tasks to be performed; an integrity level denotes associated risks and consequences for V&V

activities in case they should fail. An integrity level schema is therefore required, but its form may deviate from the one used in the standard.

The V&V processes support and must be able to address the six primary processes defined in ISO 12207 life cycle standard. These include management, acquisition, supply, development, operation, and maintenance. However, not all life cycle processes need be present for applying Std 1012.

The V&V effort must conform to all task descriptions, including their input and output, as defined in table 1 on page 27 in Std 1012-2004. The table specifies the tasks for each of the six V&V processes and defines the input and output for each of these tasks. Table 2 on page 62 in Std 1012-2004 then determines the minimum set of tasks whose completion is required for gaining a specific integrity level. In short, the standard has each V&V process (development) comprise V&V activities (requirements V&V) such that a V&V task (configuration management assessment) either is or is not applicable to it depending on the integrity level associated with the activity. The implementer may determined this by studying the tables and the related process and activity descriptions.

Std 1028, the IEEE standard for software inspections, defines five types of software review and audit, together with procedures required for executing each type. Reviews and audits are the kinds of inspection part of IEEE Std 1012 (and the larger IEEE model), as well as ISO 9000-derived QA models (ISO 9003, ISO 9126, ISO 19011.) This makes Std 1028 suitable for many kinds of application, and it seems particularly well aligned with ISO 12207 audit and review processes and/or IEEE Std 1012 inspection tasks. The inspections may be carried out by any project-associated personnel – the standard does not restrict itself to testers or developers. It is even suggested that supplier be involved if that is of benefit.

The five types of review defined are management reviews, technical reviews, inspections, walk-throughs, and audits. It is further defined that these are to be systematic, with systematic meaning team participation and documented results and procedures. How the team work is organised, is not mandated. Reviews that do not meet the standard requirements are called non-systematic; their use is not proscribed.

Management reviews monitor the progress of software products (software system comprises software products, which further comprise work products), including plans and schedules and other external documents. These reviews also evaluate existing management approaches for fitness. Technical reviews assess the software product's suitability for its intended use. Any deviations from specifications or standards are sought for. The review provides the management with information on the technical status of the project.

The third review type, inspection (not to be confused with the general, non-standard term “inspection”, with may encompass many kinds of review), aims at detecting and identifying software product anomalies. An inspection is always a peer examination, and it consists of one or more objectives and tasks, the details of which are described in IEEE Std 1028-2008 on page 16 onwards. Inspections may be of verification and/or validation type depending on the item subject to review. Walk-throughs, on the other hand, do not necessarily aim at finding anomalies. The purpose may involve many kinds of evaluation, such as locating problems, finding features that could be improved, or even educating an audience about the software product. The last review type, audit, is conducted independently, so that the review is neutral. The goal is to evaluate the software product's or processes' conformance to regulations, standards, guidelines, plans, specifications and procedures.

A detailed descriptions of each review type, as well as the procedures, participants etc. pertaining to the review, are provided in IEEE Std 1028-2008.

5.4.3 IEEE Std 1061 Software quality metrics

IEEE Std 1061-1992, Standard for Software Quality Metrics Methodology, is in many ways comparable to ISO 9126 which defined the ISO quality model and the metrics for evaluating its characteristics. Std 1061 similarly specifies the metrics for software quality assessment but does not define a specific model for their application the way ISO 9126 does. On the other hand, the way Std 1061 defines the framework for attributes and metrics can be seen as a generalised quality model, compatible for use with other IEEE standards discussed in this chapter.

Similarly to ISO 9126, the IEEE model makes use of attributes, which are measurable properties attached to the software product. Metrics, too, are employed the same way – to enable assessments of quality by enumerated values that tell the degree to which the measured attribute is achieved. There are, however, some fundamental differences as well, shown below.

The quality framework in Std 1061 comprises together with attributes quality requirements and metrics, and quality factors and their sub-factors. If the model is considered as a tree akin to ISO 9126 model, the IEEE model has the total software quality at the top with branches of quality factors beneath with direct metrics embedded in each factor. The next level contains sub-factors for each factor. At the lowest level, there are metrics for each sub-factor as the leaves of the tree.

According to this model, quality requirements are first established by assigning various quality attributes. The attributes are then assigned with user- and management-oriented quality factors. As necessary, appropriate sub-factors are assigned to these factors. The direct metrics, mentioned above, that are embedded in each factor, are quantified quality values, such as, for example, “the mean time to failure” for the factor “reliability.” The direct metrics, which can be as many as necessary, are associated with target values that when reached indicate that the factor has been achieved. The standard provides many examples on possible metrics and their target values.

The sub-factors are the result of a decomposition of each factor into software-oriented attributes, which are independent of one another and may be part of more than one factor at a time. The standard states that the sub-factors are more concrete, more technical attributes of the software, and are hence more meaningful to developers, testers, maintainers, etc. than the higher-level factors. It is hoped that the decomposition process facilities objective communication on quality between the management and the personnel.

The metrics, in turn, are the result of a similar decomposition process on the sub-factors. They are used to measure software products and processes during the software development process (life cycle.) Since it is often expensive or plain impossible to

collect values for the direct measurements early in the software life cycle, the metrics are used to estimate these values at that stage.

Factors in Std 1061 seem analogous in function to characteristics in ISO 9126. The factors' relationship to attributes is regarded differently to ISO, though, albeit this depends on how strictly the terminology is followed. Also, there is not an obvious division to internal or external characteristics – this would have to happen at the sub-factors stage if that approach is to be pursued with Std 1061. Unlike ISO 9126, the standard does not stipulate the factors or their sub-factors – the implementer is free to choose them. On the other hand, instructive examples on all three are provided as past of the Std 1061 document.

5.5 Summary

A study on open, international standards on software testing and quality assurance was conducted in order to learn of the available standards as well as how they approach their objectives and relate to one another in the larger scheme of QA. This study, together with chapter 6, formed the basis for the findings documented in chapter 7.

The chapter was divided into two main parts, ISO and IEEE standards families. In each part select individual standards, based on their perceived relevance, were reviewed in more detail and placed on the larger QA map according to their role together with other standards covered more briefly. One standard from outside ISO and IEEE, BCS 7925, was also included.

It was found that the approaches ISO and IEEE have adopted in this type of standardization are quite different. On one hand, ISO has fewer software development testing specific standards than IEEE; on the other hand, IEEE standards seem rather scattered, with a large number of individual standards that cater to a single QA issue per each so that they may be "plugged in" to a mix of standards from various sources to fill in specific needs. The IEEE standards do, however, also work together if one need not standardise all aspects of software QA. The ISO family is more limited. On the other hand, ISO 29119, which is a new standard now being developed by ISO with support from IEEE, seems very interesting in that it promises to fill in many of the gaps now present in testing and QA standardisation and so that a single, coherent view is endorsed, which current standardisation seems to be lacking with the result that the implementer needs to know the available standards and how these standards relate to one another.

6. E-BUSINESS STANDARDS AND TECHNOLOGIES

For this chapter a study on current eBusiness standards and technologies was conducted in order to determine which B2Bi techniques would seem most fitting for the requirements posed by the case projects and which, hence, would have to be considered in their development when combined with the results from chapter 5.

The chapter is divided into three parts: First, different approaches to business-to-business integration are explained; then, in the next two parts, technologies that enable B2Bi are studied with the eLive case projects viewpoint.

6.1 Business-to-business integration

All eBusiness standards and techniques considered in this chapter deal with a certain kind of business-to-business integration (B2Bi.) Although this term is somewhat vague, it is often used in place or in addition to eBusiness. Pasanen (2006) has provided a number of literature definitions for B2Bi, which show that eBusiness and B2Bi are interchangeable in the context of this thesis provided that B2G is part of B2Bi. The definition for eBusiness was given in chapter 2.

6.1.1 B2Bi approaches

Axline et al. (2002) divide B2Bi approaches into two general categories: one-to-one and hub-and-spoke (figure 10 on page 62.) In one-to-one integration, two business partners connect to one another directly with no middle-man or interaction by third parties. According to Pasanen (2006), one-to-one protocols are still the most-widely-used eBusiness approaches. This methodology was established and has been supported by the use of EDI (chapter 6.1.2). Modern eBusiness techniques allow the trading partners to deviate from this strategy (chapters 6.2 and 6.3.) The problem with this approach is that when the number of trading partners increases, the number of individual “links” that will have to be established grows rapidly, inevitably generating additional clerical costs for the participants.

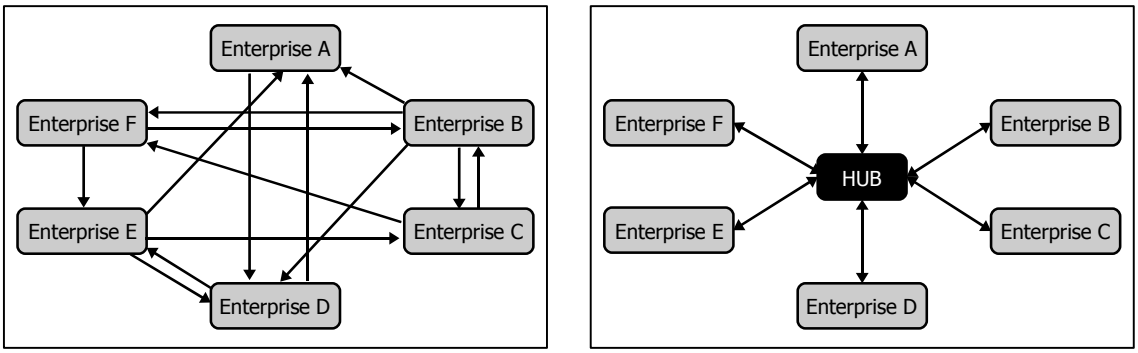

Figure 10: One-to-one (left) and hub-and-spoke B2B integration. (Pasanen 2006)

In hub-and-spoke integration, an external intermediary or hub operates between the trading partners. An arbitrary number of entities may communicate via the middle-man with an arbitrary number of other entities. In contrast to one-to-one integration, the entities involved may include not only established trading partners but also possible trading partners or even competitors. Axline (2002) emphasises that although the middle-man reduces the number of connections, which seems more efficient, the possibility of competitors sharing the same intermediary may have negative implications on adoption on the supplier side, as this arrangement allows the buyer-side to choose any supplier able to integrate with the hub. That is, the middle-man levels the playing field on the seller side.

Another way to look at B2Bi integration is a division into web services and other services, as done by Alonso et al. (2004). This view emphasises the differences in the technologies employed and the objectives set rather than the topological nature of the integration – a web service may well provide either one-to-one or many-to-many integration, or even both. Other eBusiness methods, on the other hand, are often strictly point-to-point or legacy solution, and unlike web services, employ ad-hoc technologies and make limited or no use of the open XML-based web standards studied in chapter 6.2, which often makes them platform- or application-dependent. More details on the nature of web services are provided below.

6.1.2 Web services

The number and scope of different definitions given in literature for web services is abundant. In order to retain compatibility with Alonso et al. (2004) distinction for the types of B2Bi, adhered to in this thesis, and in order to keep the definition simple, the following adaptation from a W3C definition with extensions from Wright (2005) is used: *A web service is a software application with an URI[I]-identified interface to the web. The interface may be defined, described and discovered by XML-based artifacts. The service supports direct interaction with other services by XML-based messages, which are exchanged via internet protocols and parsed and mapped to the underlying execution environment upon receipt.*

This view on web services makes a clear distinction to programming-language, application- or platform-dependent eBusiness applications. It emphasises interoperability in the form of a structured, markup-language (instead of binary) -encoded messages. It has the web service reside on the internet and so that it may be discovered and accessed in standard means by services that need it – it is not a random web application or a program run only in a local network. XML (Extensible Markup Language) is discussed in chapter 6.2.1.

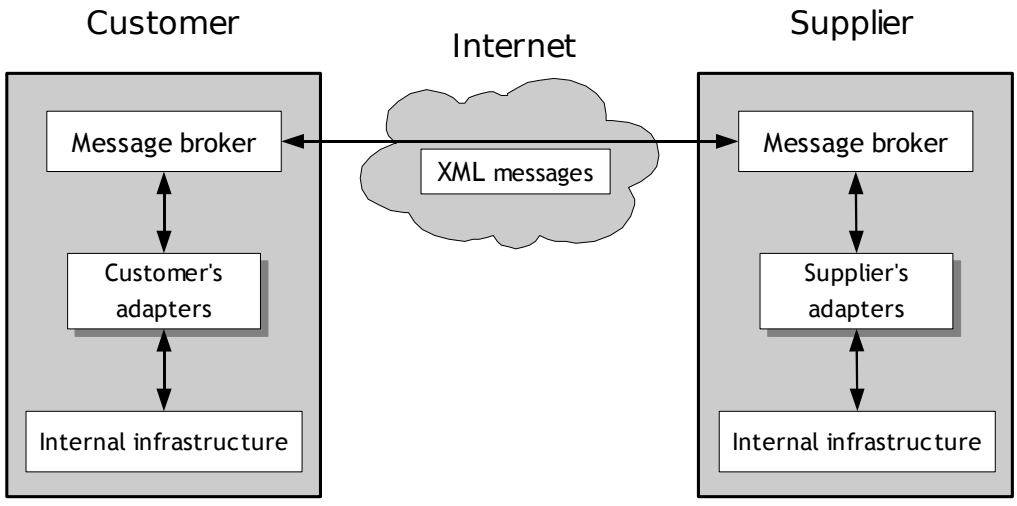

Figure 11: One-to-one web service integration. Adapted from Alonso et al. (2004)

I Uniform Resource Identifier, identifies resources on the internet. URL is a locator for an URI.

Figure 11 depicts a generalised web service integration between a customer and a supplier. A message broker receives and sends XML messages and adapters process (parse and map) the messages for the internal system.

6.1.3 EDI and EDIFACT

The EDI (Electronic Data Interchange) was the first widely-adopted B2Bi technology. It is often referred to as a single standard, but EDI is, in fact, a family of standards whose origins lie in the work conducted by the United Nations between 1960s and 1980s to release a standardised methodology for exchanging B2B documents electronically. (UNECE 2009)

Applications that make use of EDI are typically not web services, although this is possible. The EDI framework has two systems exchange binary-encoded messages via any applicable electronic means (today, internet protocols are typically used.) The format of these messages is according to the base UN EDIFACT standard, of which exists country-specific, extended versions and version tailored for the needs of particular industries. The EDI messages are composed of syntax – often according to one of the EDI standards – and elements, which carry the data for the document and which need to be agreed on prior to exchanging messages. (Salminen 1995; NIST 1996; Valtonen 2004)

Because there is no single standard for EDI, standardisation issues have hindered its adoption even for MNC's (Pasanen 2006) and overall technological- and costs-related barriers for smaller enterprises (chapter 2.) Today, EDI is still widely-deployed in the industry despite its shortcomings. However, a transition to newer, XML-based alternatives is under way. (Korpela et al. 2007) As such, there have also been efforts to map EDI standards to XML documents and to adapt it for more efficient use on the internet with the open business initiative. (Benatallah et al. 2003)

6.2 XML and Web technologies

The number of web technologies now available for building web services is greater than can be reasonably covered here. Therefore, the standards and techniques discussed in this and the following chapter have been selected according to the requirements perceived in the case projects.

6.2.1 XML

XML is a markup language. This means that an XML document consists of text and marks or tags which attach meaning to the tagged parts. The content, including the tags, is always composed of normal text and is human readable. XML is, however, meant to be consumed by software, not by humans, and its plain text format has to do with the need for platform independency. Another popular markup language is HTML (Hypertext Markup language), which is familiar from web pages where it is rendered for display by a web browser. Unlike HTML, XML has no ready-made or standardised tags; rather, XML is a meta language, where the user will have to define the tags (define an XML language, actually) themselves if a standardised structure is needed.

For simple XML documents no external definitions are required. Shown in figure 12 is a simple XML document, describing the data for a message. The first line tells the document follows an XML specification, which requires that a root element (Message)

```
<?xml version=”1.0”?>
<Message>
   <Recipient> Alice </Recipient>
   <Sender> Bob </Sender>
   <Subject> Meeting tomorrow </Subject>
   <Body> Don't forget our meeting tomorrow
        at <Underline tyoe=single> 9.30am!
        </Underline><New Line/> Last week you forgot...
   </Body>
</Message>
```

Figure 12: Simple XML document

encapsulate all other tags present. The tags then form a hierarchy where all lower level tags are part of a higher level one, forming XML containers.

For more complex data, and for specifying a standardised XML language for a set of documents, DTD (Document Type Definition) and an XML Schema can be used. The result is that when a software consumes the document, it knows, based on this specification, how to interpret the tags and attributes. The DTD or schema tells which elements and attributes may be used, what data types can be present, and how they relate to one another. This way, it is not only possible for a computer to receive and generate XML documents "on its own", fully automatically, but also to validate documents for conformance to a schema (which itself is also XML), build templates, and check documents for syntax errors. These qualities have made XML the basic building block for web service technologies. (Alonso et al. 2004; Siitonen 2004)

6.2.2 Basic web service technology

The core standards for building and using web services are WSDL (Web Service Definition Language) and SOAP (Simple Object Access Protocol). UDDI, discussed in chapter 6.2.3, provides the means for discovering such services.

WSDL is an XML-based language used for defining the interface through which a web service is accessed. WSDL has a similar role to traditional IDL's (Interface Definition Languages) used in distributed systems: It separates the abstract description of what services are offered by the interface from the exacts of how these services are implemented. Similarly, it caters to a client-server paradigm, where the service requestor is the client and the service provider is the server. The same entity may, of course, act in both roles with regard to other web services. A client may read in the WDSL definition what functionality the service offers as well as how to construct the messages (typically SOAP) for accessing this functionality.

WSDL defines the interface as ports, which are XML-abstractions for network end points. A port is a method which is bound to a concrete software implementation, hidden from the client behind the WSDL definition which is all the client needs to know

in order to operate on it. A client accesses a port by sending a message addressed to it and supplied with the appropriate parameters. By the WSDL it knows to wait for certain kinds of message as a reply. A group of related ports is said to form a service. A web service, then, comprises any number of such services. (Alonso et al. 2004; W3C WSDL 2.0 2007)

SOAP messages are the recommended language for speaking between a client and a WSDL interface, although the latter standard allows for the server to support any means of message exchange – the notion of "message" is itself an abstraction, too. SOAP follows the RPC (Remote Procedure Call) approach where a client program may cause software (its own code or other) to be executed on a remote system without knowing the explicit details on how to run in that environment. Since WSDL realises an RPC-kind approach, SOAP aligns well with it.

SOAP messages are formatted in XML and are transfer protocol and implementation independent. The SOAP message exchange methodology is one-way and stateless, meaning that the user will have to embed the interaction logic into the data carried by the messages or into a pre-defined protocol known to both parties. For lower level transport protocols, such as the synchronous internet protocol HTTP, the SOAP specification provides a ready-made binding. A SOAP message can be regarded as a container inside which data are passed from the sender to the receiver. The message consists of two main blocks, SOAP header and SOAP body, both wrapped inside a SOAP envelope (figure 13 on page 68.) The main information to be passed is put in the body block while the header block, which is not mandatory unless it is needed, should hold such data as are needed for processing the message en route (for example, a security-related intermediary may need this kind of data.) Both blocks may hold additional sub-blocks. (Alonso et al. 2004; W3C SOAP 1.2 2007)

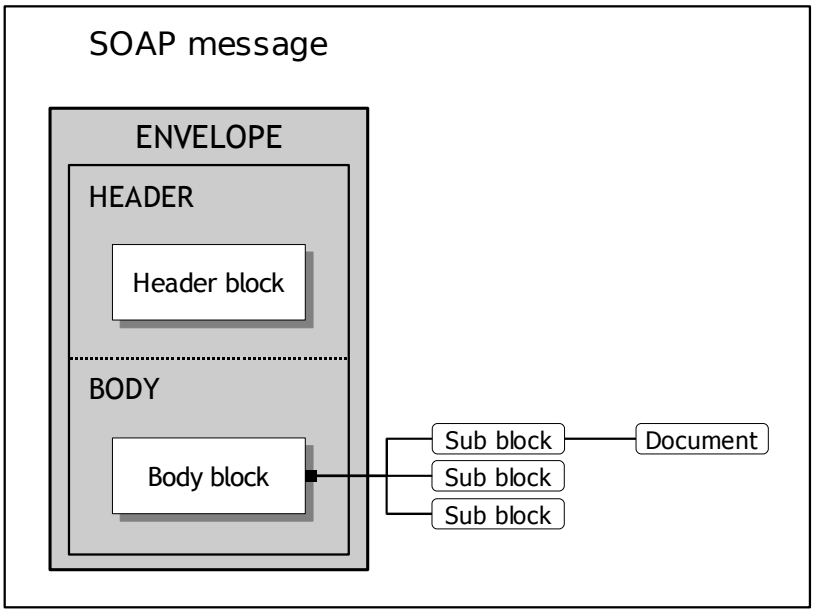

Figure 13: SOAP message structure

6.2.3 Service and information discovery

WSDL is able to define the service interface for a client to use, but the problem of how to find these definitions is not addressed by WSDL itself. This is why the web service framework includes a third component, registry, which the client (requestor) may use to discover service providers.

UDDI (Universal Description Discovery and Integration) is an open standard for describing and discovering web services in XML-based registries. UDDI provides three different kinds of registry: White pages for companies' contact information, yellow pages for categorising businesses by standard taxonomies, and green pages for detailing how a service is to be invoked. The registries employ a data structure where different characteristics are described in special tModels (technical model); a characteristic could be a WSDL, a classification, a free-form description of what the service does, and so forth. In the registry, a set of tModels is then referenced by the registry-type general description. The same tModel may be referenced by more than one service.

The UDDI specification defines APIs (Application Programming Interface) for UDDI inquiries and for publishing in UDDI registries. Specialised UDDI tools also exists.

ebXML registries (Chapter 6.3.1) are quite similar in purpose to UDDI. (Alonso et al. 2004; OASIS UDDI 3.0.2 2004)

Where registries help in discovering services and learning about them, the concept of meta data provides the means for categorising and describing pieces of data. Meta data – which is information about data – might tell who authored a web page, for example, or what time a photo was taken, who took it, where copies of it are found and to what categories it belongs. In a way, registries, too, are meta data, but about web services. Meta data is in the centre of semantic web, a tern that covers various approaches to automating the presentation and handling of semantic data. The idea is that all aspects from service discovery to finding and using the relevant data could be automated by tools that are able to consume meta data in standardised formats. This makes finding and using data more efficient. Two efforts for realising the potential of semantic web for web services are briefly covered.

RDF (Resource Description Framework) is a family of standards for processing meta data about resources on the web. The RDF data model is a triple. This means that in RDF one composes statements about objects so that the statement includes three components, each of which may have an arbitrary value. The components are subject, predicate and object. The following description, "Company A is located in Edinburgh", could be presented as an RDF triple such that *Company A* is the subject, *is located in* is the predicate, and *Edinburgh* is the object. This way, arbitrary entities and data, part of the web service, can be described in different ways. A collection of RDF statements may be represented as a graph. RDF Schema extends RDF by adding object-oriented-like relationships to the descriptions, allowing very complex meta data representations. RDF is based on XML, so it allows software, not only humans, to learn about the meaning of data, which is important for efficient information discovery. (Alonso et al. 2004; Manola & Miller 2004)

The problem with meta data is that everyone interpreting them must speak the same language. What, for example, is the meaning of Edinburgh? To make it easier to share meaning, not only descriptions, ontologies can be built for the problem domain; an example of this need is evidenced in the case projects. Ontologies are especially useful

when parties from different domains will have to share the semantic data. An ontology then determines a common vocabulary and understanding for the relevant concepts found in the domain. In practice, an ontology is often defined by concepts familiar from object-oriented design with classes, attributes, relationships and instances of classes. Ontologies may also be built from existing, semantically poorer data with various levels of automation. (Alonso et al. 2004; W3C OWL Reference 2004; Benslimane et al. 2006)

An ontology should be both fully machine-interpretable and mappable to related XML meta data in order for software to make use of it. (Bechini et al. 2008) The ontology should also be reusable across domains and services. (Benatallah et al. 2003) There exists currently a number of different approaches for specifying ontologies, including, but not limited to UML, DAML+OIL and OWL. Some notes on the possible use of OWL (Web Ontology Language) in the case projects are found in the case project descriptions and in the results in chapter 7.

6.3 eBusiness frameworks

The distinction between web services and eBusiness frameworks is sometimes vague. On one hand, both make use of XML-based techniques and are concerned with B2Bi. On the other hand, eBusiness frameworks can be used for building services that do not meet the web service definition. In short, a framework is like a collection of web technologies (chapter 6.2) and business process views thrown together in order to achieve a desired kind of B2Bi – for example, cross-industry for ebXML, and industry-specific for RosettaNet.

The name eBusiness framework, adopted from Kotinurmi et al. (2005), is used in this thesis in place of many others as it best captures the idea of enabling eBusiness without restricting to web services.

The number of such frameworks developed over the years is vast: Kotinurmi & Nurmilaakso (2004) have listed in their study 15 active general eBusiness frameworks; yet even this list is not inclusive. During the last few years, frameworks have

disappeared and new ones have been introduced. In addition, there has happened convergence between some of the surviving framework standards, and this development is expected to continue. (Kok 2005; Kotinurmi et al. 2005) Predicting the future in the field seems challenging due to the rapid development pace. For this reason, the two frameworks studied here are those most interesting from the case projects perspective: ebXML is an industry-independent alternative to RosettaNet, and has a number of interesting features. RosettaNet, on the other hand, is well-established and has been successfully used in a B2Bi project that is a predecessor to the eLive projects.

6.3.1 ebXML

ebXML (Electronic Business using eXtensible Markup Language) is a joint effort between UN/CEFACT and OASIS to develop a family of open, XML-based standards for global eBusiness. Together the standards form the ebXML eBusiness framework. A number of ebXML specifications have become part of ISO 15000. ebXML is industry-independent and specifies no business processes; the standard does, however, provide a language (BPSS) for defining ebXML-compatible process descriptions. The ebXML framework consists of three major components: The messaging service, registries and profiles, and documents.

ebMS (ebXML Message Service) specifies the ebXML message exchange protocol for business processes by using SOAP and its bindings to standard lower level transport protocols as well as other existing implementations – complemented by ebMS only when necessary – to provide security (authentication, authorisation, non-repudiation, signatures), reliable messaging (guaranteed, ordered delivery without duplicates), message exchange patterns (MEPs) for transaction monitoring, and compatibility with select web service standards. More detailed description is available in OASIS ebXML Joint Committee (2006).

For establishing business collaboration, ebXML determines a set of related standards. First, an organisation may publish its services in an ebXML registry, which is similar in purpose to an UDDI registry but contains more detailed information on the services offered. The ebXML Registry Team recommends that if UDDI is used along with

ebXML registries, one should first seek for services in UDDI and then fetch the more detailed descriptions from an ebXML registry. (Hinkelman et al. 2001) An organisation submits to the registry its CPP- (Collaboration Profile Protocol) defined business profile, which details ebXML capabilities and constraints together with supported business scenarios. The registry checks the scenarios for ebXML format compliance before making them public. Business scenarios are detailed web service descriptions. (Alonso et al. 2004), (OASIS ebXML Joint Committee 2006)

In order to use the web service (i.e. business scenario), the service requestor will first have to exchange a standardised business agreement with the service provider to negotiate the scenarios to be used as well as the ebMS protocol-related issues, such as messaging, security, contingency, etc. This agreement is called CPA (Collaboration Protocol Agreement.)

Business scenarios are analogous to business processes. The scenarios are defined by an implementation-independent BPSS (Business Process Specification Schema) language, which employs XML documents for a software-interpretable specification and UML diagrams for a visual representation. As mentioned, these definitions become part of the CPP. The UML-based descriptions depict each trading partners' role and, when complemented with sequence diagrams, show how the process shall progress. However, the exact ebXML process description is generated solely from the XML documents. (Benatallah et al. 2003; Hämäläinen 2008)

The CPP determines also the characteristics of the business documents to be exchanged. The documents can be composed of three different types of component: Core components, domain components, and business information objects. According to the core components (CC) project team, CC's are entities or information objects that when put together form a coherent, business-wise relevant concept. (CC Project Team 2009) Such concepts could include a bank account, an order, an invoice, etc. Core components for the bank account entity could then include account number, account name, account holder, and so forth. CC's are stored in the ebXML core library and are re-usable across industries. Domain components and business information objects, on the other hand, are provided by specific industries or businesses. (Benatallah et al. 2003)

Figure 14 illustrates the stages involved in invoking an ebXML partnership between two organisations.

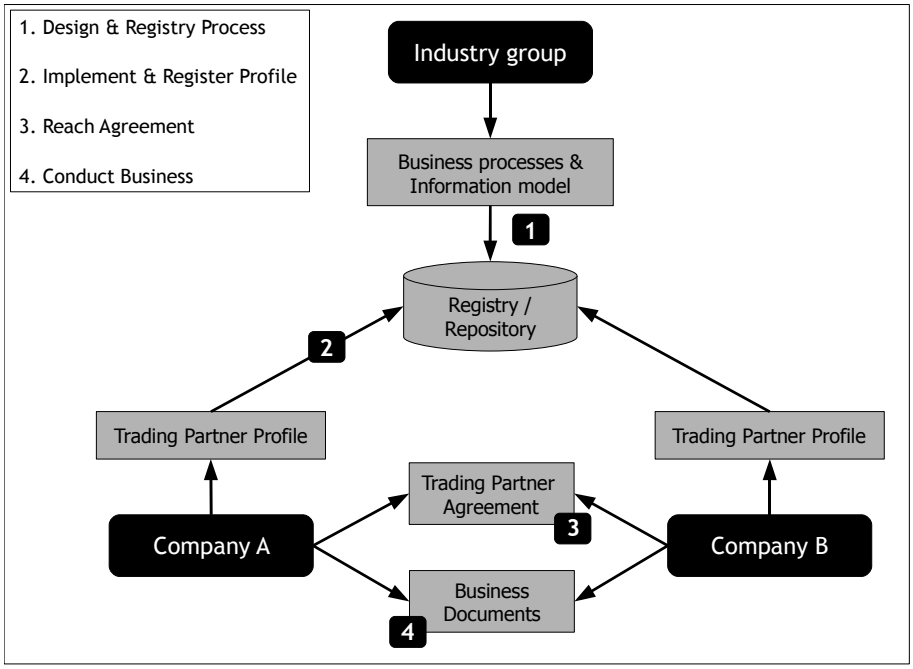

Figure 14: ebXML B2Bi invocation, high-level overview. Adapted from OASIS ebXML Joint Committee (2006)

6.3.2 RosettaNet

RosettaNet is a global non-profit business consortium with members from electronic components, information technology, and semiconductor manufacturing industries. (RosettaNet 2009) RosettaNet develops and deploys open standards for facilitating trading relationships between supply chain partners who can use the internet to exchange business documents. (Alonso et al. 2004)

Unlike ebXML, RosettaNet is therefore not a general-purpose B2Bi framework. Rather, it concentrates on the needs posed by its member industries and managing eBusiness scenarios there as well as possible. Although RosettaNet's success is difficult to quantify reliably (Kotinurmi et al. 2005), there are indications of it amongst large companies. (Damodaran 2004; Nelson & Shaw 2005; Hämäläinen 2008) This seems to attest to the

industry-specific B2Bi approach. Nonetheless, RosettaNet need not compete with the general-purpose ebXML model; due to its very nature, ebXML may well be implemented for use within the RosettaNet framework. Kok (2005) has demonstrated areas where RosettaNet could indeed benefit from leveraging ebXML standards.

RosettaNet is concerned with standardising and automating organisations' public business processes, which are those visible to trading partners. RosettaNet specifies the processes by PIP's (Partner Interface Process.) The PIP standardises business documents, the sequence of sending these documents, and the physical attributes of the messages that define the quality of service. (Damodaran 2004)

A critical aspect in integrating by a PIP is that the partners need to understand each other's terminology and day-to-day business reference codes. Also, the terminology in PIP's need to be unambiguous. For this purpose, RosettaNet provides dictionaries that standardise the common set of properties for PIP's as well as general product and partner codes. (Alonso et al. 2004)

RNIF (RosettaNet Implementation Framework) is the infrastructure for transporting PIP messages. It specifies the packaging, routing and the transport of PIP and related signal messages (acknowledgements and errors.) In addition, RNIF defines the message-level mechanisms for security (signatures and encryption) and reliability. (Damodaran 2004)

In order for business partners to integrate by RosettaNet PIP's, they will first have to define the PIP's used. This sounds simple enough, but actually involves a considerable amount of manual work and requires certain level of expertise on RosettaNet. Also, the PIP's need to be mapped to the back-end systems. These are some of the factors affecting RosettaNet adoption at the SME level.

An attempt by the consortium to address these issues in the SME-space has been an effort called RAE (RosettaNet Automated Enablement,) which, in short, tries to side-step the costs- and expertise-related issues for the SME by having the larger and more capable trading partner host a portal on which specific TRIP-PF (Trading Partner Implementation Requirements Presentation Format) forms, which are defined by plain

XML schemas – mappable to other B2B standards if necessary – convert messages between a TRIP-PIP (Trading Partner Implementation Requirements) and PDF-presented fill-in forms, which the client accessing the portal may easily consume. A TRIP-PIP is a restricted version of a RosettaNet PIP. (Cartwright 2004) As of writing, independent data on the success of the RAE effort seems to be lacking.

6.4 Case projects

The two case software development projects subject of this study are eYellowpages and eCatalogue. The projects have as their objective to build on-line services that remove some of the key barriers now preventing small- and mid-size enterprise from adopting the kind of eBusiness technologies that are needed for SME's to exchange business documents fully electronically with already-eBusiness-capable large and multinational companies and public sector organisations as well as other SME's. It has been found that the currently available eBusiness frameworks, developed for and employed by large organisations, require such resources and expertise as the average SME does not have.

Of the two projects eYellowpages is reviewed here in more detail, as there existed at the time of writing more information and more concrete feature plans for it due its starting earlier than eCatalogue, which is introduced more briefly.

6.4.1 Background

The case projects are part of a national-level eLive initiative whose goal is to promote independent research on ICT and eBusiness use in the SME sector in order to find new and more efficient ways for SME's to conduct business. The case projects are managed by Lappeenranta Innovation Ltd, which is a non-profit company owned by the city of Lappeenranta in Finland. eBusiness project manager Mr. Kari Korpela from Lappeenranta Innovation acted as the company representative and the eBusiness domain expert for this thesis. Korpela has managed and contributed to a number of research and implementation projects on eBusiness adoption in regional, national and EU-wide contexts; the case projects base on and add to this research.

6.4.2 eYellowpages

The project has as its goal to develop an information service that brings together and solves a number of related issues on B2Bi enablement. The service has the following three main objectives: To make it easy for businesses to discover potential trading partners and to learn of the partner's B2Bi capabilities; to make setting up integrated

business processes easier between partners at various stages of B2Bi readiness; and, building on these, to make SME-to-MNC and SME-to-government B2Bi more feasible.

The service is implemented as a “yellow pages” information registry. This is planned to be a web service, accessible through an open interface and API (Application Programming Interface) to any client (web services, applications, etc.) The service provides an organisation (which need not be an enterprise) with an electronic address at which all information about the organisation can be discovered. This information includes not only traditional free-form descriptions but also descriptions of the offered products and services in machine-interpretable format together with the organisation's current B2Bi capability. These data are supplemented with all necessary information to connect and set up electronic business processes with the given organisation. This entails providing interface descriptions (WSDL's, for examples), message format descriptions (SOAP's, XML schemas, etc.), and other pertinent information. An organisation need not, however, be eBusiness capable to have presence in or to make use of the service. Additionally, the service will integrate the national KATSO authentication system for identity management, authentication and authorisation; because the KATSO system is already in use in a number of eGovernment initiatives, where it is integrated with the national business information system, organisations need not create or manage different credentials for accessing the eYellowpages service. This also allows eYellowpages to act as a centralised authenticator for clients operating through the service.

Due to the early stage at which the project was at the moment of writing, exact specifications and requirements were not yet available. However, a number of features are known to be planned. These include, in addition the information service capabilities listed above, increasing both the volume and the use of semantic data generated in the service with subsequent versions. This way, the data stored are interpreted and understood by the service, and the information registry would be built as an ontology-based, interlinked network of information storage, updated and built in real time by a service agent operating in the background and triggered by activities taking place in the yellow pages domain. This kind of use of semantic data would allow some of the

envisaged capabilities: For example, the service would be able to find potential partners for a client based on an understanding of the client's requirements, capabilities and business area, which would be evaluated against other member organisations. Another planned feature is one where the client could request bids through the yellow pages service so that all potential suppliers, even those not yet known to the buyer, will be involved; or the client could simply ask for a listing of partners it has not yet been in contact with but which would seem potential. A related concept is a "buoy:" A client may drop a buoy in the eYellowpages "sea," where, depending on its type, the buoy may act on the client's behalf by listening to or broadcasting messages, or by waiting for trigger events to occur for it to react to or inform of.

The service will leverage semantics also in "encouraging" clients to become more eBusiness ready. If a client is not capable of processing electronic product data or bids, for example, it can still receive notifications and possible invitations from potential partners through the service, showing the company the business opportunities adopting certain capability would offer. That the service will also attempt to aid in common tasks so as to prevent mistakes ("This is not the company you should be sending that invoice to"), is hoped to act as another impetus for less experienced clients to start digitalising their business document handling.

These kinds of capability, based on the power of a system-wide, real-time ontology, could be realised by RDF and OWL type technologies. It is not yet clear whether existing standards such as UDDI or ebXML registries would be used as part of the service infrastructure, although ebXML registries (chapter 6.3.1) in particular seem well-aligned.

Finally, with the open API any client may integrate the service transparently with their back-end systems or other software which may need to consume these data as part of the client's business processes. For clients unable to use the API, a web interface, accessible by a browser, will be made available.

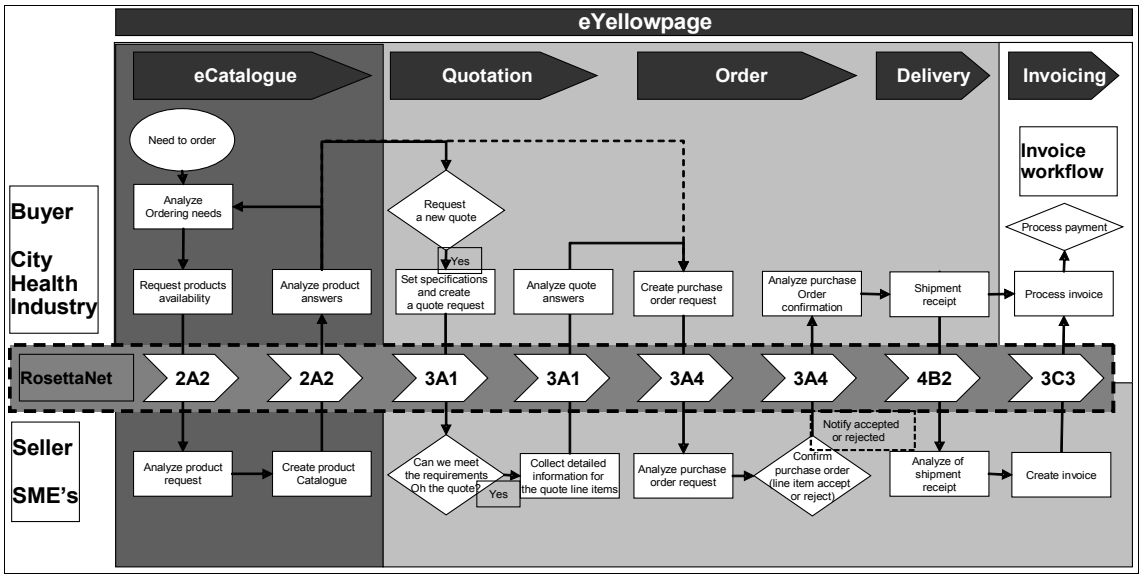

Figure 15: RosettaNet business process view to eLive projects. (Korpela 2009)

6.4.3 eCatalogue

eYellowpages facilitates service discovery and message exchange, but does not provide such detailed technical product information as is required to initiate a procurement process or request bids from potential suppliers. eCatalogue project aims at developing a web service where these product data elements required for such business processes are stored and made available through an open interface for clients (SME's, MNC's, government, etc.) to bring, update and consume. The service is also accessed as necessary by the eYellowpages web service so that information there can be linked with the appropriate product data in the catalogue so as to make the fetching of an existing or a potential partner's product data elements more automated. Figure 16 on page 80 illustrates the eCatalogue service concept used between two trading partners as part of their B2Bi. That is, although the two services integrate and make use of each other's resources, they will remain accessible as individual services.

The status of eCatalogue in March 2009 was that the project was expected to start in fall 2009; because of this, only few implementation details were yet available and only a rather general description could be provided.

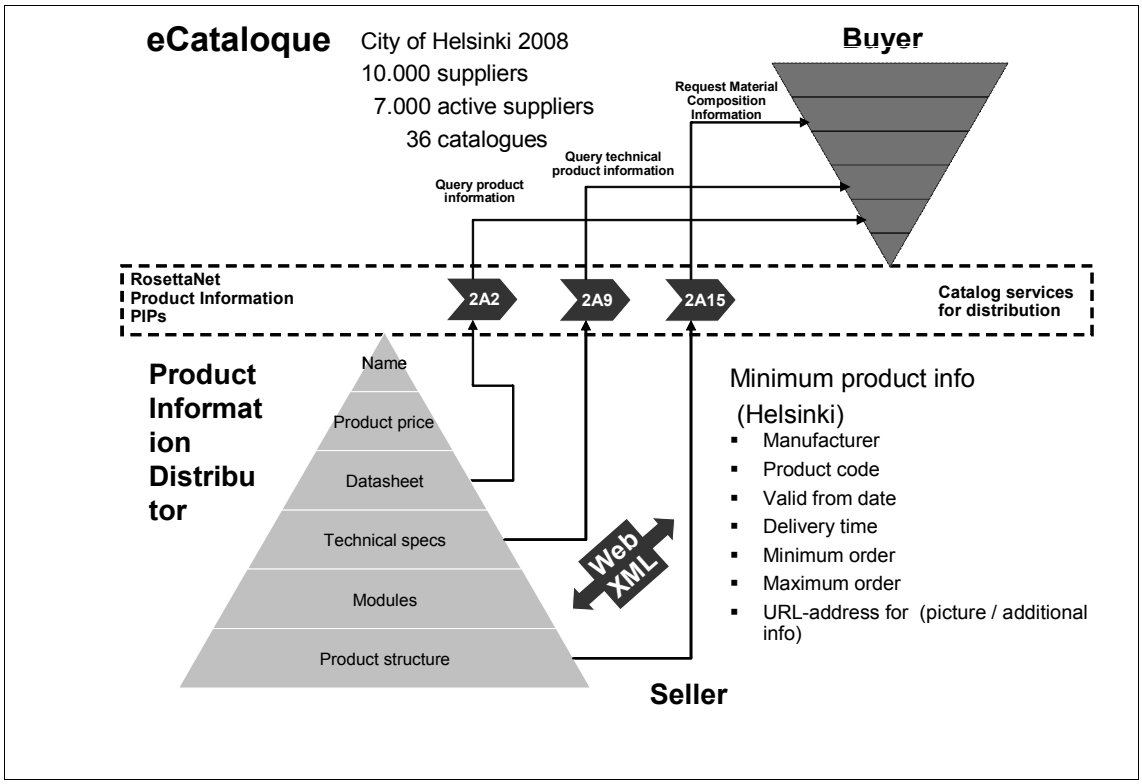

Figure 16: eCatalogue as part of a RosettaNet-based B2Bi. Korpela (2009)

6.5 Summary

The chapter provided an overview of the key eBusiness standards and technologies now available for building software for B2Bi. The techniques included were those considered most important for the requirements perceived in the case projects, which were also introduced in this chapter.

Two views to B2Bi were presented – division to one-to-one and many-to-many integration, and division to web services and other services – by which it was found that the case projects would, based the information available, have to adopt a web service approach. A number of the key XML-based web technologies required for building web services were reviewed. These techniques do not, however, restrict themselves to be used only for web services, as was noted in the study on ebXML and RosettaNet eBusiness frameworks, which make use of a number of standard web technologies in addition to their own XML-based techniques so as to enable a business-process-level B2Bi between organisations – for example, between a supplier and a buyer.

The findings on the applicability of select eBusiness standards for the development of the case projects are presented in Results, in chapter 7.

7. QUALITY STANDARDS AND E-BUSINESS

The question this thesis strived to answer was what standards and techniques on eBusiness and software testing and quality assurance would best fit for the development of the eLive software projects considering their novel approach for building software for B2Bi enablement.

7.1 Standards on QA and B2Bi

The literature review conducted showed that little beneficial, differentiating properties can be observed in comparing the available open, international standards on testing and QA when applied to building software with the case projects' requirements. Standards are developed so that they are as widely and universally applicable as possible, yet very definite and unyielding in characteristics that need to be evaluated for conformance or quantified for measurement. The conclusion was that the existing standards seem neither preclude nor enable any such software development process related characteristic that would, with the current knowledge of the case projects, make a given standard or a selection of standards evidently better fitted for developing these software projects.

The study on the existing eBusiness standards necessitated that the requirements posed by the case projects were assessed as best possible with the preliminary specifications made available. This indicated that the use of web technologies is indeed required in order to build the necessary capability into the software; the key technologies are discussed in chapter 6 and in the case project description, where it was concluded that the software would be best implemented by adopting the web service concept, defined in chapter 6.1.2. This is mandated by the need for open interfaces that serve a heterogeneous client base. The web service concept also lends to the use of existing XML-based web technologies such as WSDL and SOAP which are now well-established and mature. Some of the more advanced features proposed for the software, based on the automated use of semantic data in various ways, would require the use of RDF- and OWL-like technologies to interpret and manage a service-wide ontology.

Compared to the web service approach, there seems to be little benefit in trying to use eBusiness frameworks directly as the basis for these services. On the other hand, the use of ebXML technologies, such as registries, or RosettaNet business processes in parts of the services or in inter-service communication, may well turn out feasible.

7.2 Automated testing

The study on the B2Bi technologies together with the requirements perceived in the case projects led to the main finding of this thesis: The special requirements for testing do not lie in the software development process itself; rather, it was realised that in order for the eLive web services to provide the designed functionality, they would have to be able to carry out automated testing on eBusiness standards as part of their operation – an essential design aspect that had not yet been considered in the project documentation.

It also seems that for the services to be successfully further developed by adding new features that consume increasingly complex semantic data in sophisticated, new ways, an automated testing framework for various kinds of regression and integration testing would have to be deployed.

7.2.1 Rationale for automated testing

The need for the eLive services to carry out automated testing is mandated by two primary factors: First, they are web services that interact with a heterogeneous client base; literature already establishes this as a potential requirement for automated testing (chapter 7.2.5), and it seems that this need is realised for the eLive projects; secondly, even though the services are open for a client to insert, and within certain limits to modify, various kinds of XML-based data, the requirement for format conformance for all resident data – which may be consumed by another client or parts of the service – have to be extremely strict to protect service integrity, to allow the automated use of semantic data, and to ensure that no client will receive incorrectly formatted data from another client through these services. In practice, none of these quality requirements can

be met by manual testing. The kinds of automated testing the services will have to carry out can be divided into interoperability and conformance testing.

7.2.2 Automated interoperability testing

The purpose of automated interoperability testing is to ensure that the client desiring to access the service adheres to the web service protocol. This is required because the services are accessible through an open interface for any client software. Interoperability testing for both eCatalogue and eYellowpages would involve "a battery" of automated tests run against each connecting client prior to granting them access. For a service like eCatalogue, it may be essential to additionally prove that any required service-specific security features (encryption, digital signatures, fingerprints, etc.) are conformed to. eYellowpages has similar needs to eCatalogue due to the open API which any party is free to integrate to their software and which hence cannot guarantee that the client software implements the API correctly.

7.2.3 Automated conformance testing

Message conformance ensures that messages are syntactically correct. This kind of testing is also part of interoperability testing, but here it is considered in the context where documents (messages) are exchanged when the client already has gained access to the service, i.e. the software is "inside" the web service.

ebXML registries are required to automatically test all business process descriptions received for ebXML format conformance before making them available. This way, it can be made sure that no client will receive non-standard descriptions from the registry. A similar need is evident in both eLive services: For eCatalogue, it is paramount that the data elements the clients bring are first of all according to a validated schema, and secondly, according to the schema they claim and other clients accessing the catalogue expect. In case this were not ensured, the service could not guarantee interoperability. Similarly, once a client is accepted in the eYellowpages service, any descriptions, interface definitions (WSDL's) and documents they bring or modify will have to be automatically checked for validity.

With eYellowpages, the need to ensure data integrity is essential for the faultless operation of the service itself, too. It would be impossible for the service to operate on semantic data and maintain the service ontology, which enables most of its planned features, if all processable data are not guaranteed to be syntactically correct and according to a schema or RDF-kind-of description by which the service can reliably interpret these data. In an interlinked network of information entities, corrupt or wrongly formatted data could pose serious problems. Yet, since the client base is heterogeneous – meaning that an organisation may at will insert or modify their data in the service and do so by using client software whose implementation is beyond the eLive services developers' control – no confidence can be placed on the client side that documents being modified or documents coming in are according to the expected format. On the other hand, a client who fetches a business partner's data from the service will have to be able to trust that these data are syntactically correct and according to the claimed definition. Furthermore, it is obvious that this testing cannot be manual for a web service; nor can it be avoided by forcing all access through a web-browser-only interface or a specific client software, as the very requirements for these services mandate that they are fully open web services, available for B2Bi.

7.2.4 Implications for the software development process

The need for interoperability and conformance testing imposes that the web technologies that will be adopted will have to support the approach where all data has to be verifiable. It is therefore suggested that the services demand that all data are XML-based and according to a schema. All schemas, then, should be according to base or root schemas, defined by the service itself, with all changes to them subjected to regression testing within a testing framework deployed as part of the QA process. This approach avoids a situation where future development would be impeded by the fact that there are data stored in the service whose validity in relation to any changes made in the service cannot be verified.

7.2.5 Status of research on automated testing for web services

Research on automated interoperability and conformance testing for web services seems still very limited, although the need for it is acknowledged by e.g. Papastergiou et al. 2008; OASIS IIC TC 2004, and Bertolino 2007, amongst others. Papastergiou et al. (2008) note that the need for this kind of testing is increasing rapidly, but no mature test case definition languages or testing frameworks seem to exist; they suggest an extension to an XML-based, machine interpretable test case definition language called XRT and provide a simple process for defining such test cases. Their paper provides a good overview of the current status of automated web service testing.

Of the approaches available, one of the most mature seems to be ebXML IIC (Interoperability, Integration and Conformance) testing framework. It is, however, limited to ebXML implementations and is therefore unsuitable for general web service testing; parts of it (such as conformance testing for ebXML registries) might, nonetheless, be interesting from the case projects viewpoint. A different kind of approach is provided by Web Service Interoperability Organization (WS-I), which has published implementation guidelines for select web technologies in WS-I Basic Profile (now at version 1.0), which aims at making it easier to develop interoperable web services by specifying implementation requirements. WS-I provides a tool set that can be used to verify that the service conforms to the Basic Profile requirements. This approach does not, however, solve the complex testing needs posed by the case projects. Similar approaches to the WS-I tools are also provided by some testing companies who offer to test web service and eBusiness frameworks for interoperability and conformance.

Due to the early stage of research on this kind of testing, it is impossible to conclude which automated testing method or methods would best server the needs of the eLive projects. To understand how the testing requirements are best met and what testing techniques, such as test case definition languages and processes, would have to be developed, more research is needed on automated interoperability and conformance testing for web services.

8. SUMMARY

For the thesis a literature review was conducted in order to find what standards and techniques on eBusiness and software testing and quality assurance would best fit for the development of the case software projects, considering their novel approach for building software for B2Bi enablement. The two case software projects, eYellowpages and eCatalogue, had as their objective to develop on-line services that remove some of the key barriers now present for small and mid-size enterprises to adopt the kind of eBusiness technologies that are needed for them to exchange business documents fully electronically with already-eBusiness-capable large and multinational companies and public sector organisations. These services are necessary because the existing eBusiness frameworks, developed for and employed by large organisations, require when implemented such resources and expertise as the average SME does not have.

The research on standards on testing and quality assurance showed that the currently available standards do not possess characteristics that would make a select standard or a collection of standards evidently better fitted for developing the case projects. Standards are devised so that they are as widely and universally applicable as possible, yet very definite and unyielding in characteristics that need to be evaluated for conformance or quantified for measurement.

The study on eBusiness standards and technologies suggested that the case projects would be best built as web services; this conclusion was based on evaluating the documented as well as the perceived requirements the software would have to meet. The research also suggested that in order for the services to realise a number of key features that seem to require an ability to consume semantic data in different ways, an ontology-based approach might have to be adopted. The web service concept together with the suggested standards and technologies were defined in the thesis.

The above results led to the main finding of the thesis: The special QA requirements to be solved by testing do not lie in the eLive software development process itself; rather, it was realised that in order for the web services to provide the designed functionality,

they would have to be able to carry out automated testing on eBusiness standards as part of their operation. This finding was considered essential for two reasons. First, this design aspect had not yet been assessed in the project documentation even though it seemed that this single requirement would dictate all other aspects of the software development process and hence the selection of the web technologies to be used, as they would have to enable, and preferably facilitate, automated interoperability and conformance testing for web service software. The second reason was that the finding prompted a study on the available means for building such automated testing capability into web service architectures. This showed that research on automated testing methods for web service was lacking even though the need for it had been acknowledged in literature. The conclusion was that more research on this kind of testing is needed to further the development of web services that have high interoperability requirements.

REFERENCES

Alonso Gustavo, Casati Fabio, Kuno Harumi, Machijaru Vijay; Web Services Concepts, Architectures and Applications; Springer-Verlag, Berlin, 2004.

Amghar Youssef , Benslimane Sidi Mohammed, Benslimane Djamal, Saliah-Hassane Hamadou, Malki Mimoun; Acquiring owl ontologies from data-intensive web sites; ACM International Conference Proceeding Series Vol. 263, Proceedings of the 6th international conference on Web engineering, p. 361 – 368, ACM, 2006.

Andam Ruth Zorayda; e-Commerce and e-Business, e-ASEAN Task force, UNDP-APDIP, 2003, Available online at: http://www.apdip.net/publications/iespprimers/eprimer-ecom.pdf

Axline Sheryl, Edberg Dana, Markus M. Lynne, Petrie David; The Future of Enterprise Integration: Strategic and Technical Issues in External Systems Integration; Competing in the Information Age: Align in the Sand; Oxford University Press, 2002. Available online at: http://www.bentley.edu/mlynnemarkus/documents/future-of-enterprise-integration.pdf

Balijepally VenuGopal, Nerur Sidhar; Theoretical reflections on agile development methodologies; Emergency response information systems: emerging trends and technologies, Volume 50, Issue 3 (March 2007); Communications of the ACM, 2007.

BCS SIGIST; Standard for Software Component Testing, Working Draft 3.4, 27th April 2001.

Bechini Alessio, Tomasi Andrea, Viotto Jacopo; Enabling Ontology-based Document Classification and Management in ebXML Registries; SAC '08: Proceedings of the 2008 ACM symposium on applied computing, March 2008, ACM.

Beheshti M. Hooshang, Salehi-Sangari Esmail; The benefits of e-business adoption: an empirical study of Swedish SMEs; Service Business, Volume 1, Number 3 / September 2007; p. 233-245; Springer Berlin / Heidelberg, 2007.

Benatallah Bonalem, Bouguettaya Athman, Elmagarmid Ahmed K., Medjahed Brahim Ngu, Anne H. H.; Business-to-business interactions: issues and enabling technologies; The VLDB Journal — The International Journal on Very Large Data Bases, Volume 12, Issue 1, May 2003, Springer-Verlag New York.

Bertolino Antonia; Software Testing Research: Achievements, Challenges, Dreams; International Conference on Software Engineering, 2007 Future of Software Engineering, p. 85-103; IEEE Computer Society, Washington, DC, 2007.

Bogataj Kristina, Pucihar Andreja; Paper Living Lab – eInvoicing Activities, case: Slovenia; Information Technology for Adoption and Intelligent Design for E-Government 2007, 2007.

Boh Fong Wai, Soh Christina, Yeo Steven; Standards development and diffusion: A case study of RosettaNet; Communications of the ACM, Volume 50, Issue 12; ACM, 2007.

Budgen David; Software Design; Pearson Education Limited, Edinburgh Gate, 2003.

Cartwright Jhon; RosettaNet Automated Enablement, Material Composition Workshop, Aug 30 2004; RosettaNet, 2004.

Damodaran Suresh; B2B integration over the Internet with XML: RosettaNet successes and challenges; Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, p. 188-195, ACM New York, 2004.

Deschoolmeester Dirk, Vanpoucke Evelyne, Willaert Peter; Driver and Barriers for E-Business: Evolution Over Time and Comparison Between SMEs and Large Companies; Building the E-Service Society, Volume 146/2004, Springer Boston, 2006.

DIN; Economic benefits of standardization, Summary of results, Final report and practical examples; DIN German Institute for Standardization; Beuth Verlag, 2000.

Du Hongmei, Wu Mingxuan, Zhang Li; A Study For Understanding E-commerce Adoption in China's Service SMEs From Web Usability Perspective; Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on, 2007.

Ensio Antero, Korhonen Maritta, Mykkänen Juha, Porrasmaa Jari, Tuomainen Tuula; Tietojärjestelmien standardointityön organisointi ja kehittäminen terveydenhuollossa: nykytila ja toimenpide-ehdotukset; Standardoimistyön loppuraportti; Osaavien keskusten verkoston julkaisu 3/2005, 2005. Available online at: http://sty.stakes.fi/NR/rdonlyres/00BDBB83-AEE0-467B-949C-EB73DFE12704/1302/osve3_05.pdf

Ernst Michael D., Saff David; An experimental evaluation of continuous testing during development; Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis, p. 76-85; ACM, New York, 2004.

CEN; About us, 2009. Available online at: http://www.cen.eu/cenorm/aboutus/index.asp.

Feng P.; Studying Standardization: A Review of the Literature. Proc. of the 3rd conference on Standardization and Innovation in Information Technology (SIIT2003); IEEE, 2003.

Fillis Ian, Johansson Ulf, Wagner Beverly; Factors impacting on e-Business adoption and development in the smaller firm; International Journal of Entrepreneurial Behaviour & Research, Volume 10, Issue 3; 178 – 191; Emerald Group Pulishing Limited.

Frank Steven J.; Can you patent an industry standard?; Spectrum, Volume 39, Issue 3, March 2002; IEEE JNL, 2002.

Gallagher S.; The Complementary Role of Dominant Designs and Industry Standards; Engineering Management, Volume 54, Issue 2, May 2007; IEEE JNL, 2007.

Gruber H., Körner C., Plösch R., Schiffer S.; Tool support for ISO 14598 based code quality assessments; Johannes Kepler University Lintz; Sixth International Conference on the Quality of Information and Communication Technology, 2007.

Haikala Ilkka, Märijärvi Jukka; Ohjelmistotuotanto; Talentum, Helsinki, 2004.

Hanseth Ole, Monteiro Eric; Understanding Information Structure; University of Oslo, 1998. Available online at: http://heim.ifi.uio.no/~oleha/Publications/

Hinkelman Scott, MacRoibeaird Sean, Thomas Manes Anne, McKee Barbara; Using UDDI to find ebXML Reg/Reps; ebXML Registry Project Team, 2001. Available online at: http://www.ebxml.org/specs/rrUDDI.pdf

Hämäläinen Joonas; Liitetietojen hallinta sähköisessä liiketoiminnassa; Master's thesis; Lappeenranta University of Technology, 2008.

Hämäläinen Olli; Laadullinen tutkimus ohjelmistotestauksen tehokkuuteen vaikuttavista tekijöistä; Master's Thesis; Lappeenranta University of Technology, 2006.

IEEE / ANSI; IEEE Std 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology, 1990.

IEEE / ANSI; IEEE Std 730-1998 IEEE Standard for Software Quality Assurance Plans, 1998.

IEEE / ANSI; IEEE Std 829-2008 IEEE Standard for Software and System Test Documentation, 2008.

IEEE / ANSI; IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, 1998.

IEEE / ANSI; IEEE Std 1008-1987 IEEE Standard for Software Unit Testing, reaffirmed 2002.

IEEE / ANSI; IEEE Std 1012-2004 IEEE Standard for Software Verification and Validation, 2004

IEEE / ANSI; IEEE 1016-1998 IEEE Recommended Practice for Software Design Descriptions, 1998.

IEEE / ANSI; IEEE Std 1028-2008 IEEE Standard for Software Reviews and Audits, 2008.

IEEE / ANSI; IEEE 1044-1993 IEEE Standard Classification for Software Anomalies, 1993.

IEEE / ANSI; IEEE 1061-1998 IEEE Standard for a software quality metrics methodology, 1990.

IEEE / ANSI; IEEE 1074-1997 IEEE Standard for Developing Software Life Cycle Processes, 1998.

IEEE / ANSI; IEEE 1233-1998 IEEE Guide for Developing System Requirements Specifications, 1998.

ISO / IEC; ISO Std 9126 FDIS version 10.9, 2.10.2008

ISO / IEC; ISO Std 12207:2008 Software Life Cycle Processes, 2008.

ISO / IEC; ISO Std 29119 Part 1 Draft, 16.12.2008

ISO / IEC; ISO Std 29119 Part 2 Working Draft, 22.8.2008.

ISO / IEC; ISO Std 29119:2010 Part 3 Final.

ISO; How Are ISO Standards Developed?, 2009. Available online at: http://www.iso.org/iso/standards_development/processes_and_procedures/how_are_standards_developed.htm

Jantavongso Suttisak, Sugianto Ly Fie; eBusiness Adoption Studies Focusing on Thai SME's; Clayton School of Information Technology, 2006.

Jäntti Marko; Difficulties in Managing Software Problems and Defects; Doctoral dissertation; University of Kuopio, 2008.

Kit Edward; Software testing in the real world : improving the process; ACM Press, 1995.

Klein G.O.; Standardization of health informatics – results and challenges; Yearbook of Medical Information 2002; Centre for Health Telematics, Karolinska Institute, 2002.

Kok Adrian; B2B Standards Convergence Between RosettaNet and ebXML, XML Conference, IDEAlliance, p. 1-9, 2005.

Korpela Kari; Interview at Lappeenranta Innovation Ltd., 17th March, 2009.

Korpela et al.; Paper Living Lab – Final Report D5.2:5; Information Technology for Adoption and Intelligent Design for E-Government, 2007.

Korpela et al.; Elive osaprojektin kuvaus: eCatalogue, 9th February 2009.

Korpela et al.; Elive osaprojektin kuvaus: eYellowpages, 10th March 2009.

Kotinurmi Paavo, Nurmilaakso Juha-Miikka; A review of XML-based supply-chain integration; Production Planning & Control, Volume 15, Issue 6 September, p. 608 – 621, 2004.

Kotinurmi Paavo, Nurmilaakso Juha-Miikka, Laesvuori Hannu; XML-Based e-Business Frameworks and Standardization. Computer Standards & Interfaces, Volume 28, Issue 5, June 2006, p. 585-599, Elsevier 2006.

Legner Christina, Vogel Tobias; Leveraging Web Services for Implementing Vertical Industry Standards: A Model for Service-Based Interoperability; Electronic Markets, Volume 18, Issue 1 (Feb 2008); Routledge, 2008.

Liu Chang; Platform-independent and tool-neutral test descriptions for automated software testing; Proceedings of the 22nd international conference on Software engineering, p. 713-715; ACM, New York, 2000.

Manola Frank, Miller Eric, W3C; RDF Primer; W3C Recommendation, Feb 10th 2004.

Mansikka Nina; Sähköinen liiketoiminta Etelä-Karjalan alueen pk-yrityksissä vuonna 2002; Master's Thesis; Lappeenranta University of Technology, 2002.

Murphy Andrew, Taylor Michael; SMEs and e-business; Journal of Small Business and Enterprise Development; Volume 11, Number 3, p. 280-289, 2004.

National Institute of Technology (NIST); Electronic Data Interchange (EDI); FIPS-PUB 161-2, 1996.

O'Donnell Jon, Oksala Steven, Rutkowski Anthony, Springer Michael; The Structure of IT Standardization; StandardView, Volume 4, Issue 1, March 1996; ACM, New York, NY, USA, 1996.

OASIS Core Components Team; ebXML Core Components, 2009. Available online at: http://www.ebxml.org/project_teams/core_components/core_components.htm

OASIS ebXML Implementation, Interoperability and Conformance Technical Committee; ebXML Test Framework Committee Specification Version 1.1 DRAFT, 11 October 2004. Available online at: http://www.oasis-open.org/committees/documents.php?wg_abbrev=ebxml-iic

OASIS ebXML Joint Committee; The Framework for Ebusiness, An OASIS White Paper, 2006. Available online at: http://www.oasis-open.org/committees/download.php/17817/ebxmljc-WhitePaper-wd-r02-en.pdf

Papastergiou Spyridon, Pentafronimos Giorgos, Polemi Nineta; Interoperability testing for e-government web services; ACM International Conference Proceeding Series Vol.

351; Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance, p. 316-321, ACM, New York, 2008.

Pasanen Teemu; The Role of Third-Party Service Provider in B2B Integration; Master's thesis; Lappeenranta University of Technology, 2006.

Perry William; Effective methods for software testing; Jhon Wiley & Sons, 1995.

Pesonen Olavi; Testauksen automatisointi ja oliopohjaisten järjestelmien testaaminen; Master's thesis; University of Joensuu, 2003.

Pulli Hennariina; Factors Affecting the Adoption of Electronic Invoicing in South Karelia; Master's Thesis; Lappeenranta University of Technology, 2005.

Reid Stuart; The New International Software Testing Standard; presentation, 2008.

Rogers M. Everett; Diffusion of Innovations; The Free Press, 1995.

Rogers M. Everett, Scott L. Karyn; The Diffusion of Innovations Model and Outreach from the National Network of Libraries of Medicine to Native American Communities; University of New Mexico, 1997. Available online at: http://nnlm.gov/archive/pnr/eval/rogers.html

RosettaNet; About RosettaNet, 2009. Available online at: http://www.rosettanet.org/cms/sites/RosettaNet/About/index.html

Salminen Airi; EDIFACT for business computers: has it succeeded?; StandardView, Volume 3, Issue 1, March 1995; ACM, 1995.

Shailendra C., Jain Palvia, Sushil S. Sharm; E-Government and E-Governance. Definitions / Domain Framework and Status around the World; ICEG, 2007; Available online at: http://www.iceg.net/2007/books/1/1_369.pdf

Sommerville Ian; Software Engineering; Pearson Education Limited, Edinburgh Gate, 2007.

Standardit ja standardisointi, SFS-käsikirja 1; Suomen standardoimislitto (SFS), 2006.

Taipale Ossi; Observations on software Testing Practice; Doctor of science thesis; Lappeenranta University of Technology, 2007.

United nations Electronic Comission for Europe (UNECE); United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport, 2009. Available online at: http://www.unece.org/trade/untdid/welcome.htm

Valtonen Jaana; EDI-järjestelmän käyttöönottoprosessi pienessä tukkuliikkeessä; Master's thesis; Lappeenranta University of Technology, 2004.

Voutilainen Vesa, Pento Tapio; Electronic invoice processing as a tool for cost reduction; Frontiers of e-Business Research 2003, eBRC, 2003, Available online at: http://www.ebrc.fi/kuvat/215-229.pdf

Vuorensyrjä Matti, Savolainen Reijo (edit.); Tieto ja tietoyhteiskunta, Gaudeamus, 2001.

W3C Recommendation; OWL Web Ontology Language Reference, Feb 10th 2004. Availabe online at: http://www.w3.org/TR/2004/REC-owl-ref-20040210/

W3C Recommendation; SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), 2007.

W3C Recommendation; Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, 2007.

Wright Madeleine; A detailed investigation of interoperability for web services; Master's thesis; Rhodes University, 2005.