



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ FAKULTETAS

INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

Kiril Nugmanov

E.PASLAUGŲ VYSTYMO STRATEGIJOS

PUBLIC E-SERVICES DEVELOPMENT STRATEGIES

Baigiamasis magistro darbas

Informacinių technologijų studijų programa, valstybinis kodas 62407T104

Duomenų gavybos specializacija

Informatikos inžinerijos mokslo kryptis

Vilnius, 2009

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

TVIRTINU

Katedros vedėjas

(Parašas)

(Vardas, pavardė)

(Data)

Kiril Nugmanov

E.PASLAUGŲ VYSTYMO STRATEGIJOS
PUBLIC E-SERVICES DEVELOPMENT STRATEGIES

Baigiamasis magistro darbas

Informacinių technologijų studijų programa, valstybinis kodas 62407T104

Duomenų gavybos technologijų specializacija

Informatikos inžinerijos mokslo kryptis

prof. habil. dr. Genadijus Kulvietis

Vadovas prof. habil. dr. G. Kulvietis

(Moksl. laipsnis, vardas, pavardė)

(Parašas)

(Data)

Vilnius, 2009

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Technologijos.....mokslo sritis
Informatikos inžinerijos.....mokslo kryptis
Informatikos studijų kryptis
Informacinių technologijų.....studijų programa, valstybinis kodas 62407T104
Duomenų gavybos specializacija

TVIRTINU
Katedros vedėjas

(Parašas)
GENADIJUS KULVIETIS
(Vardas, pavardė)

(Data)

**BAIGIAMOJO MAGISTRO DARBO
UŽDUOTIS**

.....Nr.
Vilnius

Studentui (ei)**KIRIL NUGMANOV**.....
(Vardas, pavardė)

Baigiamojo darbo tema: **E.PASLAUGŲ VYSTYMO STRATEGIJOS**

patvirtinta 2009 m. d. dekanų potvarkiu Nr.

Baigiamojo darbo užbaigimo terminas 2009 m. gegužės 25 d.

BAIGIAMOJO DARBO UŽDUOTIS:

ATLIKTI VIEŠŲJŲ VYRIAUSYBINIŲ E.PASLAUGŲ ANALIZĘ. PAGAL KASMETINĘ EGEP PROGRAMOS, REMIAMOS EUROPOS SAJUNGOS, REZULTATŲ ATASKAITĄ IŠTIRTI LIETUVOS RESPUBLIKOS VYRIAUSYBĖS TEIKIAMAS VIEŠĄSIAS E-PASLAUGAS. ATLIKTI VEIKLOS ANALIZĘ BEI SUFORMULUOTI E.PASLAUGŲ PAKLAUSOS PRIEŽASTIS. IŠTIRTI ESAMUS TECHNOLOGINIUS SPRENDIMUS ŠIOJE SRITYJE BEI JŲ VYSTYMO STRATEGIJAS. REMIANTIS ATLIKTA TECHNOLOGIJŲ ANALIZE PARINKTI TINKAMIAUSIĄ SPRENDIMĄ INTEROPERABILUMO PROBLEMAI SPREŠTI VALSTYBINIO SOCIALINIO DRAUDIMO FONDO VALDYBOJE PRIE SOCIALINĖS APSAUGOS IR DARBO MINISTERIJOS. ĮGYVENDINTI ATLIKTOS ANALIZĖS REZULTATUS EDAS SISTEMOJE.....

Baigiamojo darbo rengimo konsultantai: PROF. HABIL. DR. GENADIJUS KULVIETIS

(Moksl. laipsnis, vardas, pavardė)

Vadovas
(Parašas)

... PROF. HABIL. DR. GENADIJUS KULVIETIS
(Moksl. laipsnis, vardas, pavardė)

Užduotį gavau

.....
(Parašas)

..... **KIRIL NUGMANOV**.....
(Vardas, pavardė)

.....2009.01.01.....
(Data)

Vilniaus Gedimino technikos universitetas

Fundamentinių Mokslų fakultetas

Informacinių technologijų katedra

ISBN ISSN

Egz. sk. 2

Data 2009-05-11

Duomenų gavybos studijų programos baigiamasis magistro darbas

Pavadinimas **E.paslaugų vystymo strategijos**

Autorius **Kiril Nugmanov**

Vadovas **prof. habil. dr. Genadijus Kulvietis**

Kalba

☐

lietuvių

☒

užsienio

Anotacija

Baigiamajame magistro darbe nagrinėjamos e.paslaugų plėtros tendencijos ir strategijos, aprašomi esminiai technologijų, taikomų sistemų integracijai ir interoperabilumui užtikrinti, skirtumai. Taip pat išanalizuota Europos Sąjungos vyriausybinių institucijų pletojamų projektų eGep dinamika. Analizuojant nustatyta, kad tarp rinkoje siūlomų integracijos technologinių sprendimų yra keli ryškūs lyderiai.

Kaip tinkamiausias į paslaugas orientuotos architektūros technologinis sprendimas, įgyvendinantis vieną iš eGep programų, buvo pasirinkta Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos SD formų apdorojimo sistema EDAS.

Įvairių tarpinių taikomųjų programų serverių priemonių (middleware), įgyvendinančių į paslaugas orientuotą architektūrą, gausa verčia pagrindinius rinkos dalyvius naudoti savo sprendimuose naujausias technologijas. Įmonės, turinčios didžiausią rinkos dalį, siūlo tokius programinius sprendimus, kaip Oracle SOA Suite, IBM WebSphere Business Integration Server Foundation, SAP NetWeaver ir Microsoft BizTalk Server.

Programinės priemonės pasirinkimas labai priklauso nuo naudojamos programinės aplinkos ir įmonių taikomų architektūrinių sprendimų. Dėl to Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos EDAS sistemai buvo parinktos Oracle SOA Suite tarpinių taikomųjų programų serverių priemonės (middleware), kurios buvo pritaikytos tiek veiklos logikai, tiek e.paslaugų infrastruktūrai įgyvendinti.

Analizės metu priimti projektiniai sprendimai buvo sėkmingai įgyvendinti EDAS sistemoje.

Darbą sudaro 7 dalys: įvadas, veiklos analizė, technologijų analizė, SOA pagrindu realizuotų sistemų analizė, Valstybinio socialinio draudimo fondo valdyboje prie Socialinės apsaugos ir darbo ministerijos įdiegta EDAS sistema, išvados ir siūlymai, literatūros sąrašas.

Darbo apimtis – 83 p. teksto be priedų, 19 iliustr., 2 lent., 50 bibliografiniai šaltiniai.

Atskirai pridedami darbo priedai.

Prasminiai žodžiai: SOA, tinklinės sąsajos, wsse, WS-*, e.paslaugos, integracija, interoperabilumas, JEE, ESB, BPEL.

Vilnius Gediminas Technical University
Fundamental sciences faculty
Information technologies department

ISBN ISSN
 Copies No. 2
 Date 2009-05-11

Data mining study programme bachelor (master) thesis.

Title: **Public e-Services Development Strategies**

Author **Kiril Nugmanov** Academic supervisor **prof. habil. doc. Genadijus Kulvietis**

Thesis language

☐ Lithuanian

☒ Foreign (English)

Annotation

This final master's work analyses e-services development trends and strategies, describes the fundamentals of various technologies used for the system interoperation/integration. Also are analyzed dynamics of development of the European Union government agencies projects from the eGep interoperability programme. Results of analysis states that among proposed products are several notable leaders.

SOA was chosen as the most suitable technological solution implementing one of the eGep project in Lithuania – Online Representation of State Social Insurance Board under the Ministry of Social Security and Labour – EDAS.

Variety of middleware products implementing SOA approach forces main market players to implement latest technological solution in their products. Companies with the largest market share offering following software solutions: Oracle SOA Suite, IBM WebSphere Business Integration Server, SAP NetWeaver and Microsoft BizTalk Server.

Choice of software solution strongly influenced by the currently using software and business environment. Therefore, for the Online Representation of State Social Insurance Board under the Ministry of Social Security and Labour was decided to implement Oracle SOA Suite middleware as for business as for integration layers.

Results of analysis has been successfully implemented in EDAS system (See Appendix 1).

Structure: introduction, operational analysis, analysis of technologies, SOA based systems analysis, EDAS – online representation of State of Social Insurance under the Ministry of Social Security and Labour, conclusions and suggestions, references.

Thesis consists of: 83 p. text without appendixes, 19 pictures, 2 tables, 50 bibliographical entries.

Appendixes included.

Keywords: SOA, web services, wsse, WS-*, e-Services, integration, interoperability, JEE, ESB, BPEL.

Table of content

1. INTRODUCTION	11
1.1. RELEVANCE OF TOPICS.....	11
1.2. SCIENTIFIC NOVELTY	11
1.3. THEORETICAL AND PRACTICAL SIGNIFICANCE	11
1.4. GOALS AND OBJECTIVES	12
1.5. APPROBATIONS	12
2. OPERATIONAL ANALYSIS	13
2.1. INTRODUCTION	13
2.2. ON-LINE PUBLIC SERVICES	13
2.3. STATE SOCIAL INSURANCE BOARD UNDER THE MINISTRY OF SOCIAL SECURITY AND LABOUR	22
2.4. CONCLUSION.....	25
3. ANALYSIS OF TECHNOLOGIES	26
3.1. INTRODUCTION	26
3.2. BACKGROUND.....	26
3.3. OVERVIEW OF TECHNOLOGIES	26
3.4. SPREAD OF TECHNOLOGY AND IT'S USAGE ACCORDING TIOBE INDEX.....	35
3.5. DEVELOPMENT TENDENCIES	38
3.6. CONCLUSION.....	40
4. SOA BASED SYSTEMS ANALYSIS	41
4.1. INTRODUCTION	41
4.2. HISTORY	41
4.3. THEORY AND CONCEPTS	41
4.4. SOA AND OTHER TECHNOLOGIES.....	45
4.5. ARCHITECTURAL MODEL AND TECHNOLOGIES.....	48
4.6. INTEROPERABILITY AND TRUSTED WEB SERVICES.....	61
4.7. PRODUCTS FOR SOA BASED SYSTEMS.....	62
4.8. CONCLUSION.....	67
5. EDAS – ONLINE REPRESENTATION OF STATE SOCIAL INSURANCE BOARD UNDER THE MINISTRY OF SOCIAL SECURITY AND LABOUR.....	69
5.1. INTRODUCTION	69
5.2. SYSTEM OVERVIEW	69
5.3. ARCHITECTURE	70
5.4. INTEROPERABILITY LAYER.....	72
5.5. INTEROPERATING SYSTEMS.....	73
6. CONCLUSIONS AND SUGGESTIONS	74
7. REFERENCES	75

List of figures

FIGURE 1: COMPARISON BETWEEN SUPPLY AND USE OF ONLINE PUBLIC SERVICES FOR BUSINESS [1].....	15
FIGURE 2: THE HOLISTIC MEASUREMENT OF MODEL [1].	16
FIGURE 3: AN ELECTRONIC GATEWAY TO THE GOVERNMENT OF LITHUANIA.	17
FIGURE 4: THE MIGRATION DEPARTMENT, THE RESIDENTS' REGISTER SERVICE, THE PERSONALIZATION OF IDENTITY DOCUMENTS CENTRE AND THE KLAIPĖDA MUNICIPALITY ONLINE PROJECT.	18
FIGURE 5: SOCIAL CONTRIBUTION FOR EMPLOYEES [3].	19
FIGURE 6: NEXT GENERATION eGOVERNMENT VISION STRUCTURAL DIAGRAM [5].	20
FIGURE 7: FIVE MAJOR STRATEGY OF NEXT GENERATION eGOVERNMENT [5].....	21
FIGURE 8: WEB SERVICE UMBRELLA [28].....	33
FIGURE 9: TIOBE INDEX HISTORY FOR COBOL LANGUAGE [31].....	36
FIGURE 10: THE EVOLUTION OF BUSINESS [36].....	49
FIGURE 11: SERVICE ORIENTED TERMINOLOGY [36].....	51
FIGURE 12: IMPLEMENTED SERVICES [36]	52
FIGURE 13: WS-SECURITY TECHNOLOGY UMBRELLA [37].	52
FIGURE 14: ENTERPRISE SERVICE BUS MODEL [41].....	55
FIGURE 15: BPEL RELATION TO OTHER WEB SERVICE STANDARDS [43].....	61
FIGURE 16: IBM BUSINESS INTEGRATION REFERENCE ARCHITECTURE [46].....	64
FIGURE 17: TECHNICAL SPECIFICATION OF THE SYSTEM.....	70
FIGURE 18: INTEROPERABILITY LAYER DECOMPOSITION.	72
FIGURE 19: INTEROPERATING SYSTEMS AND CONNECTION INTERFACES.....	73

List of tables

TABLE 1: STAGES OF AVAILABILITY [3]..... 16

TABLE 2: TIOBE PROGRAMMING COMMUNITY INDEX FOR JANUARY 2009 [32]. 37

Acronyms

.NET – The Microsoft .NET Framework is a software framework available with several Microsoft Windows operating systems.

API – Application Programming Interface.

CRM – Customer Relationship Management.

DDL – Data Definition Language.

DML – Data Modification Language.

DySCo – Dynamic e-Service Composition.

EDI – Electronic Data Interchange refers to the structured transmission of data between organizations by electronic means.

eGep – Synthetic measurement criteria where indicators are supply, organizational and use indicators, meant to monitor the implementation of the i2010 eGovernment Action Plan.

eGovernment – refers to the use by government agencies of information technologies (such as Wide Area Networks, the Internet, and mobile computing) that have the ability to transform relations with citizens, businesses, and other arms of government [1].

EJB – Enterprise Java Beans

eServices – A suite of web-based products that will allow tax professionals and payers to conduct business with the IRS electronically. These services are available 24 hours a day, 7 days a week via the internet [2].

HTML – Hyper Text Markup Language is the predominant markup language for Web pages.

IDL – Interface Description Language.

iDTV – Integrated digital television.

LOT2 – Synthetic qualitative supply indicators, focusing on user-centricity

OASIS – Organization for the Advancement of Structured Information Standards is a global consortium that drives the development, convergence and adoption of e-business and web service standards.

PL/SQL – Procedural Language SQL – Oracle Corporation's proprietary procedural extension to the SQL database language, used in the Oracle database.

PRC – Remote Procedure Call.

RDBMS – Relational Data Base Management System.

RDF – Resource Description Framework.

SGML – Standard Generalized Markup Language.

SOAP – Simple Object Access Protocol.

SQL92 – Structural Query Language standard introduced and standardized in 1992 year.

SQS – Simple Queue Service.

TCG – Trusted Computing Group, successor to the Trusted Computing Platform Alliance, is an initiative started by AMD, Hewlett-Packard, IBM, Infineon, Intel, Microsoft, and Sun Microsystems to implement Trusted Computing.

UDDI – Universal Description, Discovery and Integration are a platform-independent, XML-based registry for businesses worldwide to list them on the Internet.

UML – Unified Modeling Language is a standardized general-purpose modeling language in the field of software engineering.

XML – Extensible Markup Language is a general-purpose specification for creating custom markup languages.

XML-RPC – Remote procedure call protocol which uses XML to encode its calls and HTTP as a transport mechanism.

1. Introduction

1.1. RELEVANCE OF TOPICS

Development of the eServices strategy becomes more and more important part of every enterprise solution: web services and SOA are playing an important role as a middleware of software interoperability. These technologies are being deployed increasingly in practice for e-business or Enterprise Application Integration.

Increasing of interest in Service Oriented Architecture is natural and normal for every business environment – such strategy enables decreasing of duration of development stage and dedicating more time for doing processes of analysis of interoperability, integrity and maintainability.

Due to the European Union's i2010 eGovernment Action Plan till the year 2010 majority of member countries shall maintain online availability for all Government institutions. At the current stage Republic of Lithuania as a member of EU has about 75 % of online availability for Government institutions. This factor indicates that Lithuania has strong potential in IT industry and it's believed that Lithuanian government will succeed in eGovernment action plan.

To ensure success in the area mentioned there is a strong need to develop strategy, architectural solutions and maintain practices of public eServices and interoperability between government institutions and business environment.

1.2. SCIENTIFIC NOVELTY

The development of public e-Services has become an important competitive concern in many service industries especially in public area. Publicity and high availability became synonyms of successful implementation of development strategy.

However these strategies remain among the least studied and understood areas in stage of analysis. As a result the critical reason for the success is the growing interest from government sector. Government of Lithuania has demonstrated in this area an innovative and non-trivial thinking – investigation of this subject should increase stage of availability of on-line services, which are crucial for the European Union's i2010 eGovernment Action Plan.

Integrity and interoperability still remain the least investigated strategy, nevertheless there is strong demand for new strategies and approaches.

1.3. THEORETICAL AND PRACTICAL SIGNIFICANCE

Practical implementation of technological approach described in this master work is implemented in EDAS (Electronic Insurers Service System) – online representation of State Social Insurance Board under the Ministry of Social Security and Labour (see **Appendix 1**).

Theoretical basis is used for implementation of architectural solution of currently being developed and/or future systems.

1.4. GOALS AND OBJECTIVES

Analysis of public eServices development strategies shall show the way of further development of interoperability and integrity in State Social Insurance Board under the Ministry of Social Security and Labour. To fulfill this goal the following essential objectives shall be achieved:

- Do the operational analysis of subject, define the importance of problem and it's existence as critical for the current IT environment
- Define the list of the widely spread technologies and the way of successful their implementation in high availability systems.
- Analyze, define and describe the most appropriate solution for integrity and interoperability within high concurrency systems.
- Results of these objectives are expected to be implemented in the interoperability layer of State Social Insurance Board under the Ministry of Social Security and Labour.

1.5. APPROBATIONS

There are several publications related to this work:

1. 11-osios Lietuvos jaunųjų mokslininkų konferencijos „Mokslas – Lietuvos ateitis“ 2008 metų teminės konferencijos Informatika straipsnių rinkinys. Vilnius, Technika 2008, p. 358.
2. 12-osios Lietuvos jaunųjų mokslininkų konferencijos „Mokslas – Lietuvos ateitis“ 2009 metų teminės konferencijos Informatika straipsnių rinkinys. (under reviewing) (see Appendix 3).

2. Operational analysis

2.1. INTRODUCTION

In this chapter the importance of on-line public services and their role in nowadays environment will be analyzed. Significant increase of importance of public e-services in governmental area has been generated by the eGep2010 project.

This project is the most important stage of transformation and creation of modern European society in electronic environment. LOT2 synthetic indicator shows that in general on-line representations of general Governmental structures in European Union reach only 50%.

Annual report of eGep shows that situation in this field is dynamic and rapidly changes: number of online representations increases, quality of services becomes more and more important for the end users, the most modern technologies like electronic signature become usual and widely spread.

2.2. ON-LINE PUBLIC SERVICES

New era of on-line technologies has emerged. Technologies like iDTV, mobile TV, advanced mobile technologies can and certainly are differentiated to meet and deliver needs and diversity of services to the end user. And governments are plotting them in eGovernment projects.

Without waiting for changes in variety of new technologies Governments are developing service delivery models, using multi-channel delivery systems through public and/or private partnership and one stop shop.

The service 'car registration' is perfect example of this rapid evolution. Variety of countries has different processes of car registration where a car dealer, insurance companies and public car registration offices make their back offices interoperable to provide common, well designed service: customer buys car and car dealer has direct online access to the registration office and to insurance company, so that customer gets a bundle of services in one place.

Process of tax declaration services no longer needs paper forms – an electronic version became intelligent service gathering data from different interoperable back offices and provides to citizens a proposition of a tax declaration which they can accept or amend.

Social security benefits, rights that are due to citizens on the basis of their specific situation as a disabled, a parent or an elderly, are in some countries more and more automated: citizens don't have to apply for them, they receive them automatically. Why should a citizen be obliged to provide five times the same information concerning his condition in order to receive different rights linked to that specific condition? Is it because the service providers are different? Interoperability of the back offices of these service providers reduces this need for redundant information provision [1].

eServices have been transformed in many cases. The original measurement framework was not designed to capture these new evolutions and thus a review of the overall framework is required.

The sophistication of online services for businesses is covered by a higher usage. Impact measurement became an important topic since a Commission “eGovernment Communication” underlined “the need for further research into the economics of eGovernment, for a better understanding of costs and assessment of benefits and performances”. The European Commission ordered a study on an eGovernment measurement framework, the “eGep project” The project developed a measurement model based on existing impact measurement models and is a useful tool for performance measurement on an organizational level. It can also be used as a valuable tool for bench-learning when used in parallel in different organizations.

In the mean time, on the 25th of April 2006, the European Commission adopted the i2010 eGovernment Action Plan (Accelerating eGovernment in Europe for the Benefit of All). The Action Plan defines five priorities:

1. **No citizen left behind** – advancing inclusion through eGovernment so that by 2010 all citizens benefit from trusted, innovative services and easy access for all;
2. **Making efficiency and effectiveness a reality** – significantly contributing, by 2010, to high user satisfaction, transparency and accountability, a lighter administrative burden and efficiency gains;
3. **Implementing high-impact key services for citizens and businesses** – by 2010, 100% of public procurement will be available electronically, with 50% actual usage, with agreement on cooperation on further high-impact online citizen services;
4. **Putting key enablers in place** – enabling citizens and businesses to benefit, by 2010, from convenient, secure and interoperable authenticated access across Europe to public services;
5. **Strengthening participation and democratic decision-making** – demonstrating, by 2010, tools for effective public debate and participation in democratic decision-making.

The Commission requested the development of benchmark indicators for each of these priorities. Currently a consultation process is taking place to address the implementation of those indicators in a measurement system.

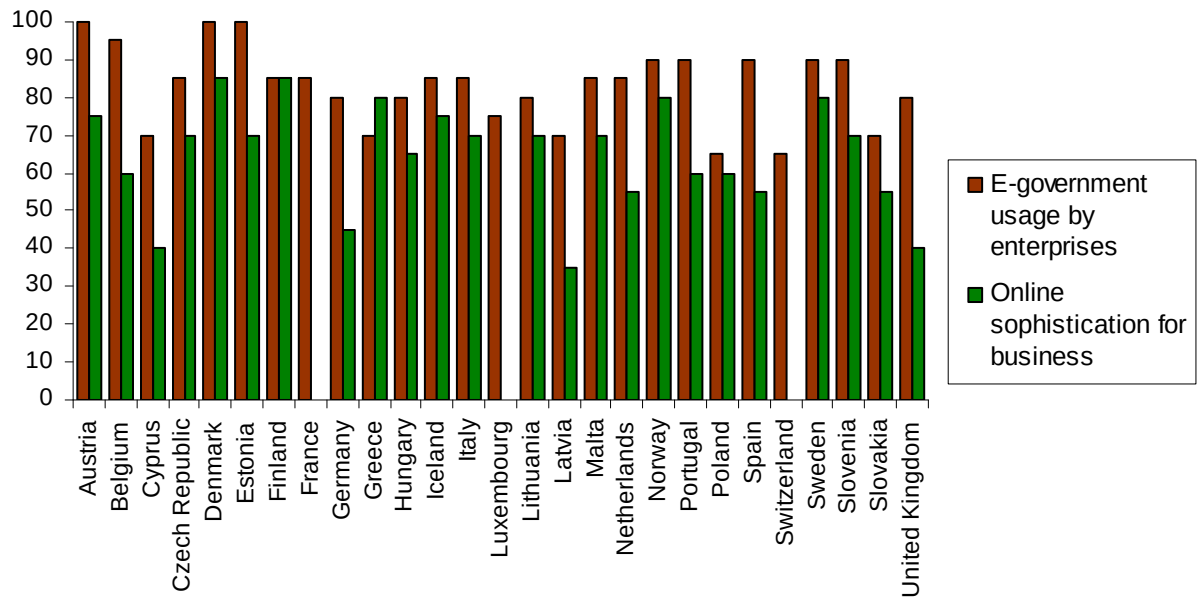


Figure 1: Comparison between supply and use of online public services for business [1].

Public services of European Union

In parallel with this 6th measurement of the availability of public services online, a pilot study “to develop and improve the eGovernment indicators” was launched. This study aims at the development and the testing of new indicators focusing on the most determinant features for take up of online services by citizens and to contribute towards achieving better and more inclusive public services. Also known as “LOT2” this study analyses existing national standards and guidelines and tries to extract some common indicators concerning accessibility and user centricity. It is expected that the findings of this pilot will feed in to future editions of this survey in some way or another.

This is indeed a complex and moving landscape that certainly will influence the format of the next edition of this study. An adapted edition of the current generic measurement model tries to put these different moves in perspective: The logic behind the model is that specifics of a nation, a region or a local environment, should be taken into consideration by studying “the structural landscape”. A readiness assessment for eGovernment programs (national, local, organizational) should cover different technical and organizational building blocks. The output should be measured as a combination of supply indicators (availability, accessibility etc.), organizational indicators (process redesign, data streamlining etc.), use indicators. A combination of those indicators can provide insights in the outcomes of the eGovernment programs [2].

The eGovernment indicator systems described above are:

- The actual availability indicators measured in this study are clearly supply output indicators

- The “LOT2” indicators will be qualitative supply indicators, focusing on user-centricity
- The “eGep” indicators are supply, organizational and use indicators, meant to monitor the implementation of the i2010 eGovernment Action Plan.
- Data from the Eurostat Households and Enterprises surveys monitor take up of online public services. The integration of these different systems in one measurement model should be the outcome of the actual European consultation process.

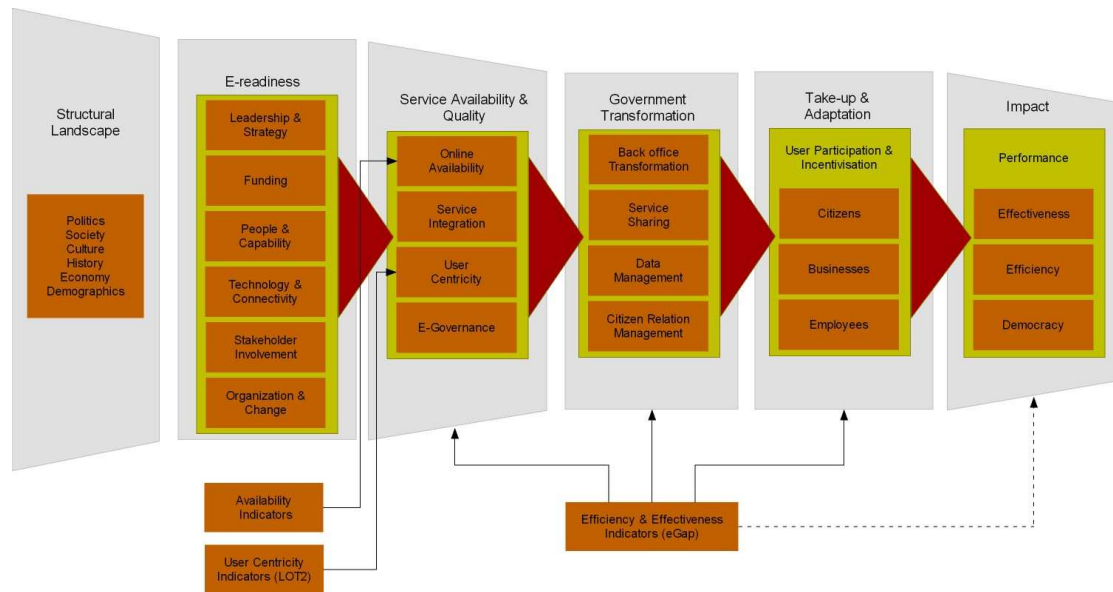


Figure 2: The holistic measurement of model [1].

Classification of stages of availability

Each eService according eGep has its own list of online availability. List of social contribution consists of 5 stages – the higher is number of stages – the greater number of possibilities user obtaining using that service. These stages are strictly described and applied all over the EU(28). Noncompliance of stages compromises whole segment of income-generating services of that member.

Table 1: Stages of availability [3].

Stage 0	The service provider or the administrative responsible level does not have a publicly accessible website or the publicly accessible website managed by the service provider or by the administrative responsible level does not qualify for any of the criteria for the stages 1 to 4.
Stage 1	The information necessary to start the procedure to declare social contributions for

	employees is available on a publicly accessible website managed by the service provider or by the administrative responsible level.
Stage 2	The publicly accessible website managed by the service provider or by the administrative responsible level offers the possibility to obtain the paper form to start the procedure to declare social contributions for employees in a non electronic way.
Stage 3	The publicly accessible website managed by the service provider or by the administrative responsible level offers the possibility of an electronic intake with an official electronic form to start the procedure to declare social contributions for employees.
Stage 4	The publicly accessible website managed by the service provider or by the administrative responsible level offers the possibility to completely treat the declaration of social contributions for employees via the website. Case handling, decision and delivery of a standard procedure to declare social contributions for employees can be treated via the web. No other formal procedure is necessary for the applicant via “paperwork”.

Public services in Lithuania

An electronic government gateway exists (<http://www.govonline.lt/>) in Lithuania. It serves as a portal to redirect citizens and businesses to the appropriate public administration website.

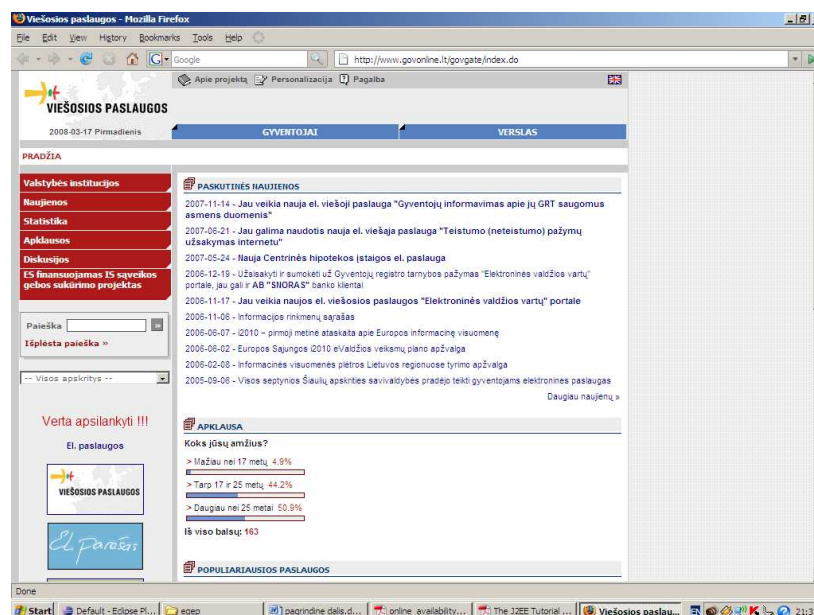


Figure 3: An electronic gateway to the government of Lithuania.

Recently the Lithuanian government started to release electronic ID cards. The government has focused more on the elaboration of an eGovernment infrastructure in the back-office. It is

expected that in the future the Lithuanian government will refocus on the development of an electronic identity card. The electronic signature infrastructure has been used to support the exchange of electronic documents in the public sector.

Passport applications are handled by local Police branches. The Ministry of the Interior, together with the Migration Department, the Residents' Register Service, the Personalization of Identity Documents Centre and the Klaipėda municipality are implementing a project on "The transfer of the service for personal documents (passport) to an electronic environment" (feasibility study), which is supported by EU Structural funds and will last until the 4th quarter of 2006.

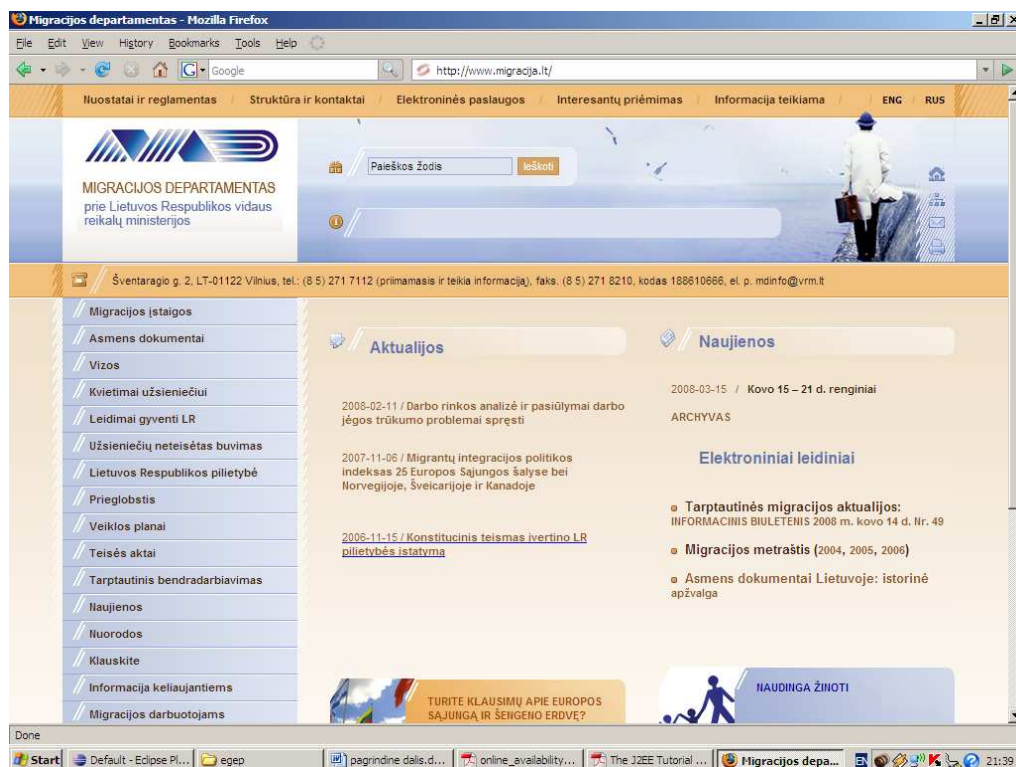


Figure 4: The Migration Department, the Residents' Register Service, the Personalization of Identity Documents Centre and the Klaipėda municipality online project.

On-line availability of social contribution services

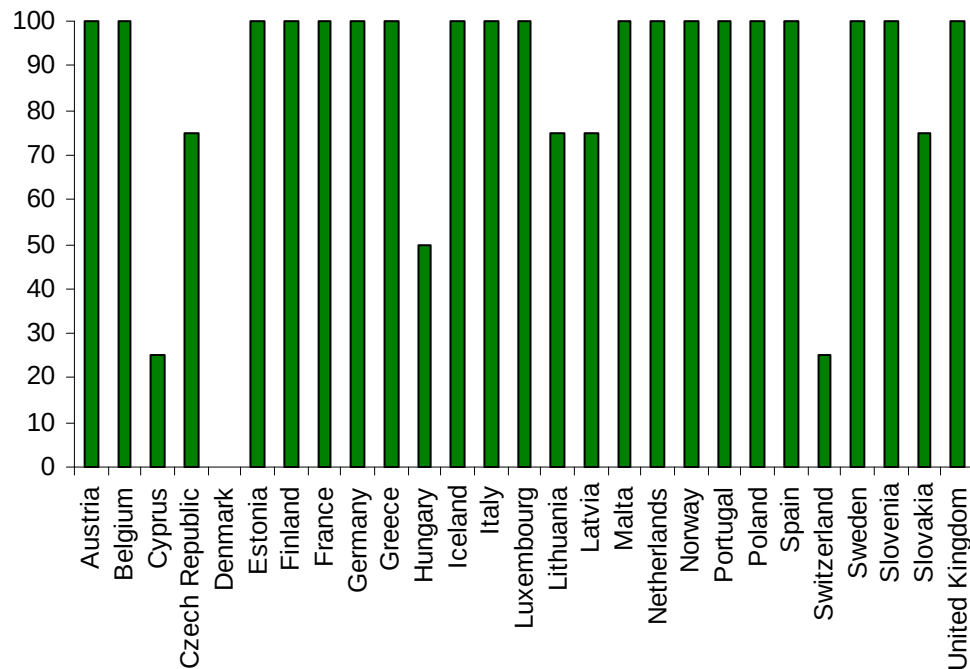


Figure 5: Social contribution for employees [3].

The different services that are offered online must be easily accessible and secure at the same time. A paragraph on eAuthentication gives an indication of the current country status. Detailed information on this subject was found in the National IDM Profiles of the Modinis-IDM website <https://www.cosic.esat.kuleuven.be/modinis-idm/twiki/bin/view.cgi/Main/NationalProfiles>.

Services are continuously evolving, so the information in this paragraph only gives an indication of the present status. Striving to offer better, easier, more complete services to their citizens, the countries have put a lot of effort in improving the offered e-services.

The latest innovative e-service developments certainly increased the country result for online sophistication. Moreover future initiatives are mostly in place to improve online sophistication and full availability online.

Next generation eGovernment

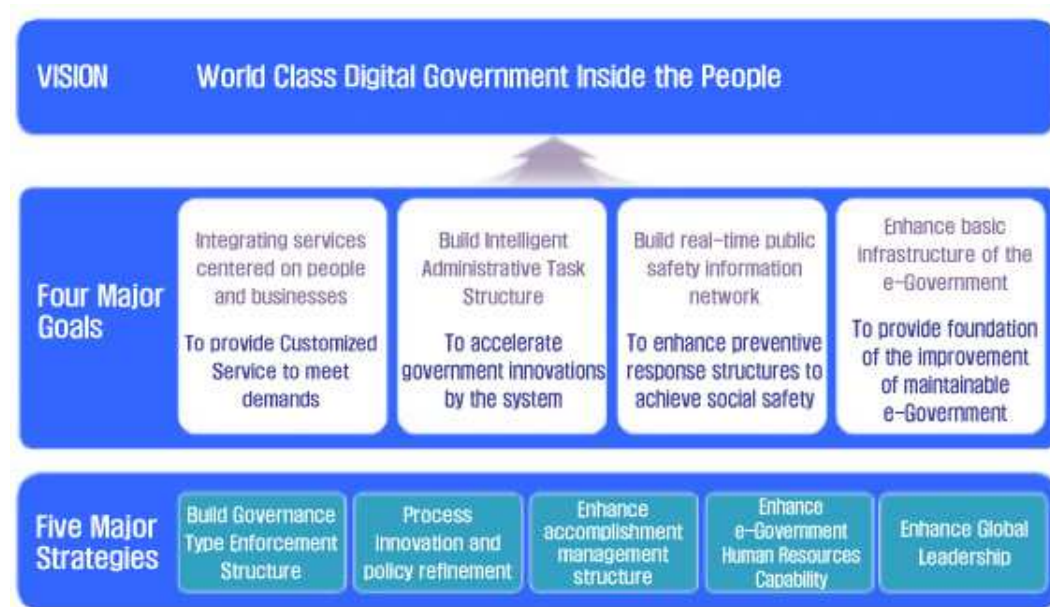


Figure 6: Next generation eGovernment vision structural diagram [5].

Next Generation eGovernment was designed to accomplish the highest levels of “*Digital Government inside the People*”. To achieve this vision, Next Generation eGovernment has four goals.

First, it aims to provide customized, integrated services that cater to the demands of people and businesses. These customized life and business information services will provide a more convenient life to citizens and more proactive support to business activities.

Second, the Next Generation eGovernment will accelerate government efficiency by building intelligent administrative systems. The core concept is to intellectualize administrative handling, through organic inter-operation between information systems; advanced collaborations between departments of central government and autonomous local bodies; and by optimizing the electronic handling of administrative tasks.

Third, Next Generation eGovernment will strengthen Preventive Response Structures, ensuring safety to its citizens by building real-time public safety information networks. These preventive public safety networks will analyze information in order to protect people from various national and social dangers.

Fourth, Next Generation eGovernment aims to maintain the continued development of basic eGovernment infrastructure. This will be accomplished by strengthening its technological and systemic foundations. All of this is in line with the idea of a Ubiquitous, technology-oriented future that will provide trustworthy eGovernment services.



Figure 7: Five major strategy of next generation eGovernment [5].

Along with aforementioned goals, there are Five Enforcement Strategies:

1. Building Governance type enforcement structure;
2. Process innovation and system examination;
3. Enhancement of performance management structure;
4. Enhancement of eGovernment human resources capability;
5. Enhancement of global leadership.

These strategies are firstly focused on establishing government-wide discussion, among the every department, and to build a governance structure where all entities of industry, academia, research, and government can be involved with shaping future implementations.

Secondly, these strategies hope to adapt policy, before the fact, to better fit an electronic administrative environment, and to improve the perspective policy's target task processing time, even before eGovernment system implementation.

Third, there will be comprehensive strategy to strengthen management structure from an achievement perspective. Each project will firmly establish goals and achievement indices before beginning, and after commencement, frequently measure achievement status, which in turn will be reflected in the budget. This transparency will bring enhanced accountability between business partners, setting contracts in step with project goals.

Fourth, there will be a strategy to enhance the career development management structure. From human resources perspective, IT manpower is needed to adapt to the new technological environment, as is producing professionals from fields such as security and business management.

Finally, the Globalization Strategy is designed to spearhead efforts to increase global respect for our world-class brand, eGovernment.

2.3. STATE SOCIAL INSURANCE BOARD UNDER THE MINISTRY OF SOCIAL SECURITY AND LABOUR

State Social insurance contributions

The State social insurance contributions as an eService are one of the income-generating services. The average of online availability in European Union is 94% [3]. If we compare scores of service within EU(10) and EU(28) we would find the indicator of online availability within EU(10) is lower. An explanation of this paradox may be that as a general rule these members are still working on their social security platform. Nevertheless the number of income-generating segment in eServices is high; the question of efficiency is still open knowing that efficiency is strongly linked to its usage all over the citizens.

Legal statements and regulations

Income-generating services and social contribution services in general are responsible for state social insurance of employees and insured. Current state of this segment is developed so that every document which has been registered in any of services (VAT, Income Taxes, and Social Insurance Contribution for Employees, Customs Declaration and Corporate Tax) must be stored for at least 75 years since their registration. This fact raises problem of storage, transporting updating and other operations with that data.

Integration of digital signature in this process solves almost all problems but in 75% of Legal Systems in countries of EU (28) don't have appropriate base regulations and legislations. In Lithuania to obtain this possibility and integrate infrastructure of digital signature in process of social contribution for employees were applied following regulations:

- *“Dėl Reikalavimų kvalifikuotus sertifikatus sudarantiems sertifikavimo paslaugų teikėjams, Reikalavimų elektroninio parašo įrangai, Kvalifikuotus sertifikatus sudarančių sertifikavimo paslaugų teikėjų registravimo tvarkos ir Elektroninio parašo priežiūros reglamento patvirtinimo” (Žin., 2003, Nr. 2-47) [6].*
- *“Elektroninio parašo ĮSTATYMAS” (Žin., 2000, Nr. 61-1827) [6].*

The similar EU directive was the following:

- “Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures” (Official Journal L: 1999-01-19 Nr.13-12) [7].

These two laws are essential for the legalization of usage of the digital signature in the Republic of Lithuania. Additional laws were the following:

- “Elektroninio parašo įstatymo 4, 8, 14, 16 straipsnių pakeitimo ir papildymo ĮSTATYMAS” (Žin., 2002, Nr.64-2572, Nr.65) [8].
- “Asmens tapatybės kortelės įstatymo 2, 4, 5 straipsnių pakeitimo bei papildymo ir įstatymo papildymo 11 straipsniu ĮSTATYMAS” (Žin., 2008, Nr. 76-3007) [9].
- “Buhalterinės apskaitos ĮSTATYMAS” (Žin., 2002, Nr. 99-3515) [10].

To implement previously mentioned changes in legal structure majority of legal acts and regulations of State Social Insurance Board under the Ministry of Social Security and Labour were approved:

From January 01, 2008 the following regulations and laws entered into force:

- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. gruodžio 20 d. įsakymas Nr. V-665 „Dėl Elektroninės draudėjų aptarnavimo sistemos taisyklių patvirtinimo“ (Žin., 2007, Nr. 139-5742)[11].
- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. gruodžio 20 d. įsakymas Nr. V-666 „Dėl Elektroninių pranešimų kvalifikuoto elektroninio parašo taisyklių patvirtinimo“ (Žin., 2007, Nr. 139-5741)[12].

Regulations considering electronic forms and related procedures also have been approve:

- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 15 d. įsakymas Nr. V-256 „Dėl 1-SD pranešimo apie apdraustųjų valstybinio socialinio draudimo pradžią ir 2-SD pranešimo apie apdraustųjų valstybinio socialinio draudimo pabaigą pateikimo taisyklių patvirtinimo“ (Žin., 2007, Nr. 68-2711) [13].
- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 26 d. įsakymas Nr. V-271 „Dėl 3C-SD pranešimo apie ūkinės bendrijos metinėje pelno mokesčio deklaracijoje nurodytos pajamų metinės sumos paskirstymą nariams pateikimo taisyklių patvirtinimo“ (Žin., 2007, Nr. 72-2879) [14].

- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 26 d. įsakymas Nr. V-273 „Dėl 6-SD pranešimo apie draudėjo reorganizavimą pateikimo taisyklių patvirtinimo“ (Žin., 2007, Nr. 72-2880) [15].
- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 15 d. įsakymas Nr. V-257 „Dėl 9-SD pranešimo apie motinai (įmotei), tėvui (įtėviui) arba vaiko globėjui suteiktas (atšauktas) atostogas vaikui prižiūrėti pateikimo taisyklių patvirtinimo“ (Žin., 2007, Nr. 68-2712) [16].
- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 20 d. įsakymas Nr. V-261 „Dėl 12-SD pranešimo apie apdraustųjų nedraudiminius laikotarpius pateikimo taisyklių patvirtinimo“ (Žin., 2007, Nr. 69-2775) [17].
- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 14 d. įsakymas Nr. V-251 „Dėl įgaliojimo pasirašyti pateikiamus socialinio draudimo pranešimus ir pranešimo apie įgaliojimo panaikinimą parengimo ir pateikimo tvarkos aprašo patvirtinimo“ (Žin., 2007, Nr. 68-2709) [18].
- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 18 d. įsakymas Nr. V-258 "Dėl Valstybinio socialinio draudimo fondo valdybos direktoriaus 2004 m. birželio 21 d. įsakymo Nr. V-249 "Dėl apdraustųjų įskaitos duomenų koregavimo" pakeitimo" (Žin., 2007, Nr. 68-2713) [19].
- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 15 d. įsakymas Nr. V-254 "Dėl Valstybinio socialinio draudimo fondo valdybos direktoriaus 2004 m. gruodžio 28 d. įsakymo Nr. V-482 "Dėl Valstybinio socialinio draudimo pažymėjimo išdavimo ir keitimo taisyklių patvirtinimo" pakeitimo" (Žin., 2007, Nr. 68-2710) [20].
- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. liepos 4 d. įsakymas Nr. V-295 „Dėl SAM pranešimo apie apdraustuosius už ataskaitinį ketvirtį pateikimo taisyklių patvirtinimo“ (Žin., 2007, Nr. 75-2998) [21].
- Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. liepos 26 d. įsakymas Nr. V-331 „Dėl SAV pranešimo

apie savarankiškai dirbančius asmenis pateikimo taisyklių patvirtinimo“” (Žin., 2007, Nr. 85-3444) [22].

2.4. CONCLUSION

This year again shows a deep gap in the performance in the public services for businesses (exceeding the two-way interaction in most countries) against those for citizens (some services still progressing in reaching the one-way interaction). Nevertheless, although both categories are showing a rise in their results, there is a slight tendency in the “old” Member States to emphasize progresses in servicing citizens (7%-point increase compared to 4%-points increase for services to businesses regarding the online sophistication in EU(18). In the new Member States, this increase is huge in both domains but maintains a similar effort invested (16%-point increase for citizens and 17%-point for businesses) [3].

These results are symptomatic of divergent concerns on where to focus the efforts as either being a country accustomed to the exercise that succeeded in “testing” its eServices capabilities with businesses and is now taking up the challenge with citizens performances or being a new member states that is going through making major progresses to aligning its eServices performances in general to those of countries being longer in the race.

3. Analysis of technologies

3.1. INTRODUCTION

The following chapter is discussing and analyzing technologies for the eServices, their reviews and essential benefits.

Rapid development of variety of technologies and fast growth of IT market created unique state of IT world – a lot of from the first look incompatible technologies reaches the same solution. Variety of technologies forces each government to make deep analysis of their economic and social situation. Corporations with unique technological solutions influence educational system to adapt to current market and its needs.

Number of specialists, spread of technology, its cost and maintenance - these are the main factors, influencing the choice of technological solution for eServices, especially in the government level.

3.2. BACKGROUND

From first using mainframe computers over 30 years ago to today's delivery of services electronically, business society has continually embraced computer technology to improve program operations and service delivery. Throughout this time, there have been notable achievements both in information technology (IT) and information management (IM).

Computers have become increasingly smaller, cheaper, and faster with more capacity. Their primary uses were processing data, then managing information to more efficiently support government ministries and, more recently, delivering service directly to the customer. IT expertise, initially the domain of IT professionals, gradually devolved to IT users, as computer literacy became more and more widespread. As the technology advanced and enabled more networking, government management, and indeed citizen and business expectations, has risen.

Changes in IT innovation were not, however, done in isolation. IT advances need to be placed in the context of the existing fiscal situations and pressures, the continuously increasing customer expectations (internally and externally), and the shifting organizational culture.

3.3. OVERVIEW OF TECHNOLOGIES

The most important anticipated benefits of e-government include more efficiency, improved services, better accessibility of public services, and more transparency and accountability [3].

While e-government is often thought of as "online government" or "Internet-based government," many non-Internet "electronic government" technologies can be used in this context. Some non-internet forms include telephone, fax, PDA, SMS text messaging, MMS, wireless networks and services, Bluetooth, CCTV, tracking systems, biometric identification, road traffic

management and regulatory enforcement, identity cards, smart cards and other NFC applications; polling station technology, TV and radio-based delivery of government services, email, online community facilities, newsgroups and electronic mailing lists, online chat, and instant messaging technologies [23].

COBOL

COBOL is one of the oldest programming languages still in active use. Its name is an acronym for COMmon Business-Oriented Language, defining its primary domain in business, finance, and administrative systems for companies and governments.

The COBOL 2002 standard includes support for object-oriented programming and other modern language features.

COBOL programs are in use globally in governmental and military agencies, in commercial enterprises, and on operating systems such as IBM's z/OS, Microsoft's Windows, and the POSIX families (Unix/Linux etc.). In 1997, the Gartner Group reported that 80% of the world's business ran on COBOL with over 200 billion lines of code in existence and with an estimated 5 billion lines of new code annually.

Near the end of the twentieth century the year 2000 problem was the focus of significant COBOL programming effort, sometimes by the same programmers who had designed the systems decades before. The particular level of effort required for COBOL code has been attributed both to the large amount of business-oriented COBOL, as COBOL is by design a business language and business applications use dates heavily, and to constructs of the COBOL language such as the PICTURE clause, which can be used to define fixed-length numeric fields, including two-digit fields for years.

SQL

Originally designed as a declarative query and data manipulation language, variations of SQL have been created by SQL database management system (DBMS) vendors that add procedural constructs, control-of-flow statements, user-defined data types, and various other language extensions. With the release of the SQL:1999 standard, many such extensions were formally adopted as part of the SQL language via the SQL Persistent Stored Modules (SQL/PSM) portion of the standard [24].

Common criticisms of SQL include a perceived lack of cross-platform portability between vendors, inappropriate handling of missing data, and unnecessarily complex and occasionally ambiguous language grammar and semantics.

SQL/OLB

The SQL/OLB, or Object Language Bindings, extension is defined by ISO/IEC 9075-10:2003. SQL/OLB defines the syntax and semantics of SQLJ, which is SQL embedded in Java. The standard also describes mechanisms to ensure binary portability of SQLJ applications, and specifies various Java packages and their contained classes.

SQL/Schemata

The SQL/Schemata, or Information and Definition Schemas, extension is defined by ISO/IEC 9075-11:2003. SQL/Schemata define the Information Schema and Definition Schema, providing a common set of tools to make SQL databases and objects self-describing. These tools include the SQL object identifier, structure and integrity constraints, security and authorization specifications, features and packages of ISO/IEC 9075, support of features provided by SQL-based DBMS implementations, SQL-based DBMS implementation information and sizing items, and the values supported by the DBMS implementations [25].

Information Engineering

Information Engineering first provided data analysis and database design techniques that could be used by database administrators and by systems analysts to develop database designs and systems based upon an understanding of the operational processing needs of organizations for the 1980s.

The Finkelstein thread evolved after 1980 into the DP-driven variant of IE. From 1983 till 1986 IE evolved further into the business-driven variant of IE, which was intended to address a rapidly changing business environment. The then Technical Director, Charles M. Richter, from 1983 to 1987, played a significant role by revamping the IE methodology as well as designing the IE software product which helped automate the IE methodology, opening the way to next generation Information Architecture.

Strategies

- **Information Strategy Planning:** The fundamental objective of Information Strategy Planning (ISP) is to develop a plan for implementing business systems to support business needs.
- **Outline Business Area Analysis:** The OBAA answers a range of questions related to implementation of a business area. Select tasks to include in a particular project that provide support for business decisions and objectives. Specific information needs and priorities for the business area are needed.
- **Detailed Business Area Analysis:** The purpose of a DBAA project is to provide detailed models as a solid basis for system design. The methodology helps find the right answers to the right questions. Applying the methodology is never an end in itself.
- **Business System Design:** The purpose of a Business System Design project is to specify all aspects of a system that are relevant to its users, in preparation for the technical design, construction, and installation of one or more closely related databases and systems. The key tasks are therefore structured to produce unambiguous consistent specifications, with the volume of detail necessary to make planning and technical design decisions.
- **Technical Design:** A Technical Design project prepares an implementation area for construction and installation. The key tasks are structured to produce a system and database that meet the user's acceptance criteria and are technically sound.
- **Construction:** The objective of the Construction stage is to produce a system, as defined in the technical specification, on time and within budget. The system should be of an acceptable quality, and contain all necessary operating and user procedures. The task is complete when the acceptance criteria for the business system are met.
- **Transition:** Transition is defined as the period during which newly developed procedures gradually replace or are interfaced with existing procedures. The execution of a Transition project obviously demands a thorough understanding of both the system to be installed and the systems to be replaced.

ORB

In distributed computing, an object request broker (ORB) is a piece of middleware software that allows programmers to make program calls from one computer to another via a network.

ORB's handle the transformation of in-process data structures to and from the byte sequence, which is transmitted over the network. This is called marshalling or serialization.

Some ORB's, such as CORBA-compliant systems, use an Interface Description Language to describe the data which is to be transmitted on remote calls.

In addition to marshalling data, ORB often exposes many more features, such as distributed transactions, directory services or real-time scheduling.

In object-oriented languages, the ORB takes the form of an object with methods enabling connection to the objects being served. After an object connects to the ORB, the methods of that object become accessible for remote invocations. The ORB requires some means of obtaining the network address of the object that has now become remote. The typical ORB also has many other methods.

OOAD

Object-oriented analysis and design (OOAD) is a software engineering approach that models a system as a group of interacting objects. Each object represents some entity of interest in the system being modeled, and is characterized by its class, its state (data elements), and its behavior. Various models can be created to show the static structure, dynamic behavior, and run-time deployment of these collaborating objects. There are a number of different notations for representing these models, such as the Unified Modeling Language (UML).

Object-oriented analysis (OOA) applies object-modeling techniques to analyze the functional requirements for a system. Object-oriented design (OOD) elaborates the analysis models to produce implementation specifications. OOA focuses on what the system does, OOD on how the system does it [26].

Java

As part of the **DySCo** program, a Java-based prototype has been developed in order to explore the concept of mediated e-service delivery. The prototype was developed in a joint effort by Hewlett-Packard Laboratories, and the University of Ferrara.

The infrastructure for **DySCo** reproduces a real-world scenario, with a number of e-service providers relying on an electronic marketplace for the offering their services. Each e-service exposes different types of information (meta-data), ranging from a functional description of the service to availability and pricing models. In order to enable mediated service delivery, we included in the meta-data information describing the customer view on the service delivery process. This information can be generated automatically, applying a role based projection algorithm to the description of the business processes implemented by the service provider. The algorithm dynamically adapts the definition of the interaction process between customer and service provider based on the roles that the customer accepts to play (e.g. receiver of the service, receiver of the invoice, quality assurance monitor). The interaction process becomes part of the service contract

between customer and service provider. The electronic marketplace mediates the interaction between the parties enforcing the processes specified in the service contract [27].

XML (common record)

Like HTML, the Extensible Markup Language (XML) was a W3C creation derived from the popular SGML that has existed since the 70's. This widely used meta-language allowed organizations to add intelligence to raw data, and exchange meta-tagged information with partner organizations.

XML was developed in response to the eBusiness movement of the late 90's, where server-side scripting languages made conducting business via the Internet viable. Through the use of XML, developers were able to attach meaning and context to any set of information transmitted across Internet protocols. Not only was XML used to represent data in a standardized manner, the language itself was used as the basis for a series of additional specifications. The XML Schema Definition Language (XSD) and the XSL Transformation Language (XSLT) were all authored using XML. These three specifications, in fact, have become key parts of the core XML technology set.

Because XML is designed to assist in information management, its functions should already be performed by records managers. Records managers should already be familiar with the features and benefits of XML through the needs of their daily work.

XML so suddenly appear because data storage and transmissions are no longer significant for most applications. Many more people are now involved in exchanging data. Considering the problems of conveying information into the future is a new initiative of many IT (Information Technology) organizations. That is, preserving data so it will be available in one, two, or three decades. (Records managers have had records series with 30 year, and even permanent retentions since the start of the profession.).

Web Services

In 2000, the W3C received a submission for the **SOAP** standard. This specification was originally designed to unify (and in some cases replace) proprietary RPC communication. The idea was for parameter data transmitted between components to be serialized into XML, transported, and then de-serialized back into its native format. Soon, corporations and software vendors began to see an increasingly large potential for advancing the state of eBusiness technology by building upon the proprietary-free Internet communications framework. This ultimately led to the idea of creating a pure, Internet-based, distributed technology. One that was not only designed specifically to function via existing Internet protocols, but one that could leverage the concept of a standardized

communications framework to bridge the enormous amount of disparity that existed between and within organizations. This concept was called Web services.

The most important part of a Web service is its public interface. It is what assigns the service an identity and enables its invocation. Therefore, one of the first initiatives in support of Web services was the Web Service Definition Language (WSDL). The W3C received the first WSDL specification submission in 2001, and has continued aggressively revising this standard. A WSDL file is the central piece of information that describes a Web service.

To further the vision of open interoperability, Web services required an Internet-friendly and XML-compliant communications format that could establish a standardized messaging framework. Although alternatives, such as *XML-RPC*, were considered, SOAP won out as the industry favorite, and remains the foremost messaging standard. In support of SOAP new role, the W3C responded by releasing newer versions of the standard to support both RPC-centric and document-centric message types. The latter are used almost exclusively within SOA.

Completing the first generation of the Web services standards family was the UDDI specification. Originally developed by UDDI.org, it was submitted to *OASIS*, which continued its development in collaboration with UDDI.org. This specification allows for the creation of standardized service description registries both within and outside of organization boundaries. UDDI provides the potential for Web services to be registered in a central location, from where they can be discovered by service requestors.

A third-party marketplace emerged promoting various utility services for sale or lease, and custom Web services were developed to accommodate a variety of specialized business requirements. Existing messaging platforms, such as messaging-oriented middleware products incorporated Web services to support SOAP in addition to other message protocols, and some organizations were able to immediately incorporate Web services to facilitate B2B data exchange requirements – often as an alternative to *EDI* solutions.

A Web service is a set of related application functions that can be programmatically invoked over the Internet. Businesses can dynamically mix and match Web services to perform complex transactions with minimal programming. Web services allow buyers and sellers all over the world to discover each other, connect dynamically, and execute transactions in real time with minimal human interaction. Web services are:

- **Self-contained** – on the client side, no additional software is required. A programming language with XML and HTTP client support is enough to get you started. On the server side, a Web server and servlet engine are required. The client and server can be implemented in different environments. It is possible to Web service enable an existing application without writing a single line of code.

- **Self-describing** – the client and server need to recognize only the format and content of request and response messages. The definition of the message format travels with the message; no external metadata repositories or code generation tools are required.
- **Modular** – simple Web services can be aggregated to form more complex Web services either by using workflow techniques or by calling lower layer Web services from a Web service implementation.
- **Platform independent** – Web services are based on a concise set of open, XML-based standards designed to promote interoperability between a Web service and clients across a variety of computing platforms and programming languages.

Web services might be anything, for example, theatre review articles, weather reports, credit checks, stock quotations, travel advisories, or airline travel reservation processes. Each of these self-contained business services is an application that can easily integrate with other services, from the same or different companies, to create a complete business process. This interoperability allows businesses to dynamically publish, discover, and bind a range of Web services through the Internet.

Technologies within the Web Services umbrella

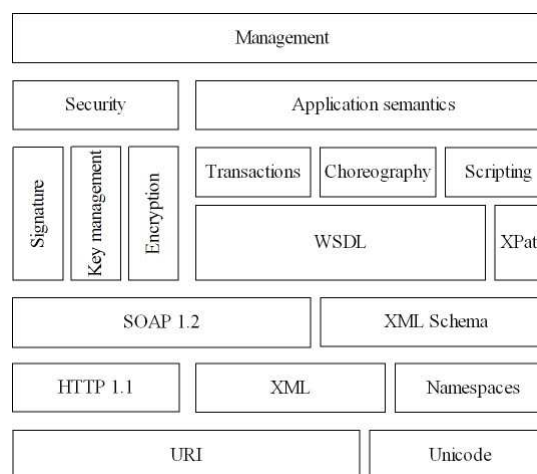


Figure 8: Web service umbrella [28]

There is a mass of different pieces being bolted onto the foundations of Web Services provided by WSDL and SOAP 1.2 and the diagram implies things considerably. The management layer is a supervisory layer allowing the control of the many agents involved in a web services-based operation. The "Application semantics" layer indicates the necessity, for any useful interoperability, to have.

Run Time messaging

The design work of web services is divided between the run time protocols and the descriptions of services.

The W3C work at runtime based on HTTP transport of XML-encoded messages, using the SOAP protocol. (Here by SOAP we mean SOAP 1.2, previous versions including early proprietary submissions which are not standards or guaranteed to interoperate). There is a bifurcation in the design at this point, as SOAP operates basically in two modes.

In one, the XML message is used to encode the parameters to a remote operation in much the same way as remote method invocation in for example, CORBA, DCOM, or RMI. In this mode, XML is used as the marshalling style, but the system is a distributed using remote procedure call in a fairly traditional way.

- There is a standard marshalling syntax
- Interfaces between software modules have well-defined functions, which in turn have well-defined and typed input and output parameters
- Stubs (dummy routines which simulate the remote procedure by a local one which communicates with the remote one) can be generated directly from the WSDL definition
- The remoteness can be transparent, making the design of a distributed system similar to the design of a program.

In the other mode, SOAP carries an XML document, and the task of the receiver is seen more as a document processing operation. This is less rigid than the RPC style.

- The interface a service provides is defined just by the XML schema. This defines the acceptable document types, which can allow extension in many ways, using XML namespaces.
- The communication is more apparent to the application writer, who deals with the document object model (DOM) of the received message, rather than having parameters that aren't marshaled automatically.
- XML tools such as XSLT and XML-Query, and XML encryption and so on can be used.
- It is simpler to use message exchange patterns other than the request/response.

The document mode of SOAP seems to be getting the most traction in the ecommerce stack. This is not an accident. The XML mode is more flexible than the RPC mode. It is easier in principle to extend an XML-based message system to include more information as a system grows. In fact, RDF is especially powerful in this area, as new information can be parsed into an entity-relationship

form by old agents, and it becomes logically clear which parts can be ignored by those who do not understand them.

Functionality which has been mentioned as required above the basic layer at runtime includes:

- Routine data within message for processing by different agents; defining workflow path of message. Black box or white box patterns of design.
- Profiling existing security technologies for use in e-business applications using web services. Authentication and key management.
- Packaging of attachments to messages. XML Packaging.
- Reliable messaging (delivery, non-duplication, ordering) for the case in which the transport layer (such as TCP under the HTTP) doesn't provide this.

3.4. SPREAD OF TECHNOLOGY AND IT'S USAGE ACCORDING TIOBE INDEX

According the COBOL published report in 2007 year there are following facts:

- There are 310 billion lines of legacy code operating in the world (65% of all software).
- Five billion lines of new COBOL code are being written every year.
- Fifteen percent of new application development is written in COBOL.
- Thirty-four percent of coding activities are in COBOL.
- Seventy-five percent of the world's business data resides on mainframe systems.
- There are 38,000 legacy systems at more than 10,000 mainframe sites worldwide.
- COBOL developers are retiring much faster than they are being replaced as educational institutions emphasize more cutting edge technologies.

Analyzing following report appeared new interesting facts as following:

- What's the trend line – is the number of new lines of COBOL code declining;
- The percent of all application development;
- And assuming it's declining, how steep is the decline.

Each fact of this report shows that adaptation and possibility of mutation of COBOL as application platform reached its own limit: COBOL developers are retiring much faster than they are being replaced as educational institutions emphasize more cutting edge technologies.

Put all statistics together will be reached a fascinating inference - it appears that COBOL doing is almost exactly half as efficient as coding in other languages. That is, the amount of coding is double the amount of development [29].

COBOL annual report was taken as an example of eService technology that was too rapidly developed and adopted for too many tasks. In current market situation there is a need of secure, stable and efficient eService technology that would be flexible for implementation various interactions between external systems. Next major need for technology is the stability and maintenance of its releases. Main market players must know the road-map of technology and its life cycle [30].

To estimate the current situation in market of eServices was taken following approach for the problem salvation: group all technological solutions by the programming language on which technology it's based.

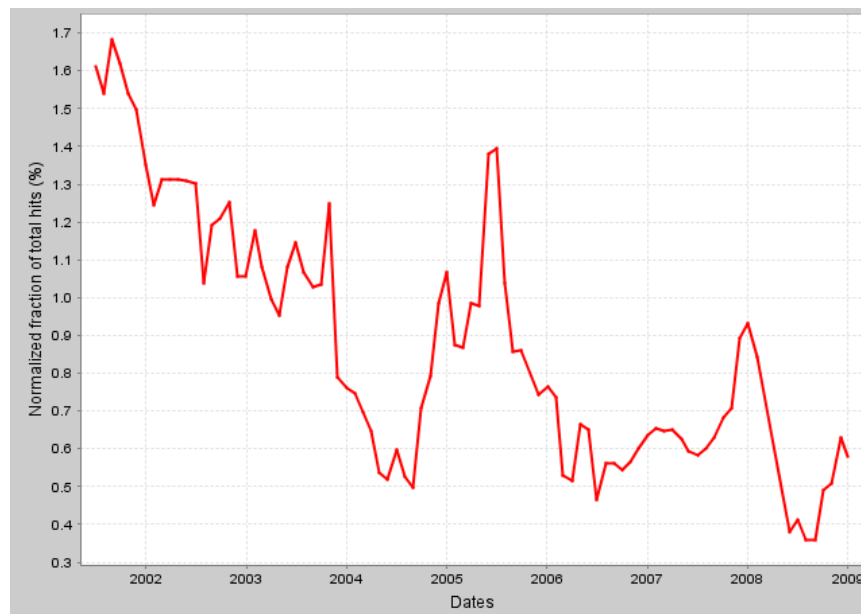


Figure 9: TIOBE¹ index history for COBOL language [31].

Putting all technology in one language dependent slice it's much more easier establish it's spread according languages popularity.

TIOBE rates all the most popular languages and indexes them. The ratings are calculated by counting hits of the most popular search engines. The search query that is used is + "<language> programming".

The search query is executed for the regular Google, Google Blogs, MSN, Yahoo!, and YouTube web search for the last 12 months. The web site www.Alexa.com has been used to determine the most popular search engines.

¹ TIOBE is specialized in assessing and tracking the quality of software, it measures the quality of a software system by applying widely accepted coding standards to it.

The number of hits determines the ratings of a language. The counted hits are normalized for each search engine for the first 50 languages. In other words, the first 50 languages together have a score of 100%. Let's define "hits50(SE)" as the sum of the number of hits for the first 50 languages for search engine SE and "hits(PL,SE)" as the number of hits for programming language PL for search engine SE. Possible false positives for a query are already filtered out in the definition of "hits(PL,SE)". This is done by using a manually determined confidence factor per query. A query such as "Basic programming" also returns pages that contain "Improve your basic programming skills in Java". The first 100 pages per search engine are checked for possible false positives and this is used to define the confidence factor. If this factor is 90%, then only 90% of the hits are used for "hits(PL,SE)". An overview of the confidence factor can be found in the groupings table below.

The ratings are calculated with the following formula:

$$((\text{hits(PL,SE1)}/\text{hits50(SE1)} + \dots + \text{hits(PL,SEn)}/\text{hits50(SEn)})/n \quad (1)$$

where n is the number of search engines used. YouTube only counts for 7%, the other search engines 23% for each.

The TIOBE Programming Community index gives an indication of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Observe that the TIOBE index is not about the best programming language or the language in which most lines of code have been written.

Table 2: TIOBE Programming Community Index for January 2009 [32].

<i>Position 2009.01</i>	<i>Position 2008.01</i>	<i>Delta in Position</i>	<i>Programming Language</i>	<i>Ratings 2009.01</i>	<i>Delta 2008.01</i>	<i>Status</i>
1	1	0	Java	19.022%	-1.83%	A
2	2	0	C	15.931%	+2.01%	A
3	5	+2	C++	10.116%	+1.39%	A
4	3	-1	(Visual) Basic	9.161%	-1.80%	A
5	4	-1	PHP	8.882%	-0.31%	A
6	8	+2	C#	5.609%	+0.75%	A
7	6	-1	Python	4.731%	-0.81%	A
8	7	-1	Perl	4.303%	-0.94%	A
9	10	+1	JavaScript	3.360%	+0.16%	A
10	9	-1	Delphi	3.303%	-0.03%	A
11	11	0	Ruby	3.149%	+0.80%	A

12	14	+2	<i>D</i>	1.022%	-0.15%	<i>A</i>
13	12	-1	<i>PL/SQL</i>	1.006%	-0.22%	<i>A</i>
14	13	-1	<i>SAS</i>	0.797%	-0.41%	<i>A</i>
15	18	+3	<i>Pascal</i>	0.661%	+0.21%	<i>B</i>
16	20	+4	<i>Logo</i>	0.632%	+0.25%	<i>B</i>
17	15	-2	<i>COBOL</i>	0.579%	-0.35%	<i>B</i>
18	28	+10	<i>ABAP</i>	0.537%	+0.34%	<i>B</i>
19	17	-2	<i>FoxPro/xBase</i>	0.477%	-0.03%	<i>B</i>
20	21	+1	<i>ActionScript</i>	0.455%	+0.11%	<i>B</i>

According TIOBE statistical results the most widely spread programming languages are following:

- Java
- C based languages (C, C++ and C#)
- SQL (background for PL/SQL based web services)²

Majority of public eServices are based on Java and C based languages thus it's natural to say that following technologies in near future will be most widely spread:

- SQL
- Java
- .NET
- XML (as common record for data exchange and transformation)
- Web services (distributed model for accessing services via HTTP protocol over SOAP)

3.5. DEVELOPMENT TENDENCIES

The nascent market for eServices will swell dramatically over the next four years, spreading well into the global arena, according to new research released today.

Radicati Group's "eServices Market 2004-2008" reports that the combined market for eServices solutions, management, integration and security will be worth \$950 million in 2004. By 2008, that figure will climb to \$6.2 billion.

A number of different factors are driving the growth of the eServices market, according to Masha Khmartseva, the report's principal analyst. A major one is that it offers companies new ways of doing business.

² Exact spread of **SQL** or **PL/SQL** as technology is unknown. 75% or all databases are using **RDBMS** and **SQL92** based **DML** and **DDL**.

Previously incompatible systems from different corporate divisions and departments can be integrated into a unified corporate portal, enabling employees to share data more quickly and efficiently. The same can be done with partner systems. Virtually any application can be exposed as an eService, which means that there is no limit to the type and kind of systems that can be integrated.

Khmartseva also noted that companies are using eServices with increasing regularity to establish communication channels with partners and customers. "We estimate that about 28 percent of all deployments are being done for external purposes," Khmartseva told www.internetnews.com.

eServices also offer significant benefits to developers, who will help fuel the growth of the market. Khmartseva noted that, from a developer perspective, eServices does not limit them anymore to any particular platform or platforms.

"eServices can be created in different environments and be ready to communicate with each other the moment they are published," Khmartseva explained. "By offering reusable components, they also cut the time spent on developing applications by up to 50 percent."

At the platform level, the eServices market is a battle between .NET and Java. According to the report, neither platform currently dominates the other.

"Today it's pretty equal," Khmartseva said. "The future will depend a lot on who will be able to offer customers better and more user-friendly tools to build, manage and integrate eServices."

That future may also see a convergence of tools for the two platforms. Khmartseva expects to see the convergence of .NET and Java platforms for eServices, manifested in more similar development tools and features that developers can expect to work with in either environment.

One of the most surprising aspects of the study is that it shows the eServices market is no longer a U.S. phenomenon, said Khmartseva.

"This is a striking difference from a year ago. Today according to our estimates, almost 48 percent of all deployments are being done outside the U.S."

The study illustrates that North American deployments lead the pack at 52 percent, with Europe following at 39 percent, Asia Pacific with 6 percent and the rest of the world accounting for 3 percent of the total.

In addition to highlighting the growing popularity of eServices and the spots for its growth, the report also noted security as a concern.

"Today eServices are mostly used to communicate non-sensitive information," Khmartseva said. "Over the next few years we'll see further development of eServices standards, especially in the area of security. This will enable companies to build more sophisticated applications, and engage in high-value, multi-transactional operations." [33].

3.6. CONCLUSION

As the Internet came into common use, the “Information Age” arrived. The Internet made electronic communication possible anytime anywhere. With increased user sophistication and computer literacy come increased expectations for the availability of information, and for services to be delivered online 24/7.

Further integration of information has been necessary to meet these expectations. Along with access, however, has been the necessity to protect and secure this information. As more and more public services are accessed electronically, policy and operational issues about privacy protection and user identification need to be addressed. Since e-services can be accessed from anywhere, the security of the network and the protection of information databases are vital.

The Internet has radically changed the service delivery model. Previously, business employees responded to the public and their inquiries or service delivery demands using technology to assist them in servicing those requests. Now, using the Internet, the public can obtain information or services directly with employees in the background managing the information and technology.

Need of fully qualified infrastructure of services forced technological progress in to the way of SOA. Orchestration and choreography of existing services became vital for any existing organization based on service providing.

4. SOA based systems analysis

4.1. INTRODUCTION

Following chapter analyzes SOA based systems, their architectural design, technologies and standards. Also software solutions for such architecture are analyzed and theory and concepts are briefly described. Important point of analysis is interoperability and integrity – IT world has variety of enterprise solutions, these technologies may differ in decades. For this reason capability to integrate with .Net technological platform becomes much less important than capability to cooperate with mainframe and COBOL based systems.

Analysis of product shows variety of implementations from different vendors such as SAP, Oracle and IBM.

4.2. HISTORY

As the interest in Web services skyrocketed, major software vendors aggressively developed a series of extensions to the first-generation standards. Known as the second-generation or WS-* standards, these specifications addressed specific areas of functionality with the overall goal of elevating the Web services technology platform to an enterprise level.

It wasn't long before organizations began to realize that instead of just accommodating existing distributed applications, Web services could become the basis of a separate platform. A platform that could realize the concept of partitioning an enterprise into a series of autonomous services. Thus, service-oriented architecture was reborn, rejuvenated with a technology platform truly capable of delivering its original goals. As stated in previous chapters, SOA is truly an evolution. Its prominence today is the result of numerous interrelated initiatives driven by a variety of standards organizations and software vendors. Through a volatile environment fueled by a mixture of collaboration and competition, standards are being strategically positioned, each defining a specific part of what we are calling the contemporary SOA technology platform [34].

4.3. THEORY AND CONCEPTS

In describing the conceptual view of Service-Oriented Architecture, one would expect a model that is representative of any and all implementations of it. This would provide the minimal concepts needed at a high enough abstraction level to facilitate a specific SOA solution, independent of the problem domain and of the technologies, standards, and implementations. This allows for growth, reuse, and interoperability as the model does not include these aspects that are ever changing. This description is often called a reference model and the Organization for the Advancement of Structured Information Standards has proposed one such model for SOA.

For example, think of a reference model for a restaurant. One imagines that a restaurant needs the concepts of an eating area, a cooking area, and a hygiene area. A specific architecture of this would entail a dining room, a kitchen, and restrooms. Supposing this restaurant also has a game room for children to enjoy, this would also qualify as a restaurant as it includes the realization of the minimal concepts needed, but has a different architecture. This shows the value that a reference model has as it can describe something in a general form to allow differing kinds of specific architectures.

The SOA reference model is concerned with seven main concepts: service, visibility, interaction, real-world effect, service description, policies and contracts, and execution context. These are grouped into three categories according to the functionality that they bring to SOA. The first one is the category of service which has the stand alone concept of service.

The second category is the dynamics of services which contains the concepts that are critical to the operation and use of the service. This contains concepts of visibility, interaction, and real-world effect. The last grouping is that of the concepts that support those in the dynamics of services. This contains the ideas of service description, policies and contracts, and execution context.

Before jumping into these concepts it is beneficial to understand the basics of how services are used. Services are provided by service providers and used by service consumers. The consumers discover a provider that can fulfill its needs through a service description and interact with it through the service's interfaces to result in a real-world effect.

While there are other factors involved, this is basic enough to understand the underlying concepts that will be detailed. It should be noted that each participant's actions are private and not known to any party outside, unless these actions affect some shared resource. The heart of service-oriented architecture is the notion of the service itself. So what is a service? Simply, a service is just a means to bring needs and capabilities together. A service allows its capabilities to be accessed through its service interfaces, but can only be used within the constraints that its policies and contracts, along with the consumer's, have specified. These interfaces only detail how to access the service's capabilities as the consumer does not need to know how it is implemented, with the exception that only enough implementation information is needed to use the service. As a result of using a service, a realization of some real-world effect occurs. This could be a result of returned information based on a request, a change to the shared state of some defined entities, or a combination of the two.

The process of going from a consumer having needs to a provider fulfilling said needs through its services is facilitated by the concepts within the dynamics of services. The first step would be for the consumer to identify a service that is capable of satisfying its needs. This leads to the concept of a service having visibility. Visibility is the basic idea that the service provider and

service consumer need to be able to “see” each other in order to interact. Consider a basic C program “Hello World” that prints a string to stdout. If you view the library `stdio` as a service and your program as the consumer, there needs to be a means for the two to know about each other. This is carried out through the use of the statement `#include <stdio.h>`. Within the concept of visibility there are three subcomponents that need to be addressed as they act as preconditions for visibility.

These are the notions of awareness, willingness, and reach ability. Awareness is the idea that one party needs to know about the existence of the other parties for an interaction to occur. This requires that a service description and set of policies be available for the consumer to know about the existence of and capabilities of the service. For the service to be known to a consumer, it may push its description and policies on a consumer, such as the case for television ads. Conversely, consumer may pull service descriptions searching for suitable services. This is analogous to a company having contractors making an offer for a certain project. Awareness is also the means to which a consumer can decide if the services capabilities match their needs.

Willingness is needed as a consumer and provider may be aware of each other, but may not be willing to interact. Suppose a consumer discovers the service and initiates an interaction, but the service refuses to cooperate and an interaction never occurs. This may be the desired behavior in the context of defending against a denial-of-service attack. The willingness to interact, however, is not to be confused with the willingness to perform some requested action. Think of basic human interaction, one may be willing to interact with their friend, but if their friend asks them to wash their car and clean their house they may reject both requests, but are still willing to interact with them.

The last aspect of visibility is the somewhat obvious one of reach ability, in that the consumer and service provider need an actual means to communicate. Without such a means an interaction is impossible as the two have no visibility of each other. Once visibility has been satisfied and the preconditions met, the next step is to actually interact with the service. This is the act of a consumer performing actions against the service.

Often times this is done through the use of message exchanges, but may also be done through other means, such as the alteration of state of a shared resource. For a consumer to understand what is entailed in interacting with the service, it must reference the service description. Primarily, it consults the service description to find out what information can be exchanged and how it may interact with the service. These models called the information model and behavior model, respectively.

The information model specifies what information may be exchanges as well as the syntax and semantics of the said information. The syntax describes that structure of the information, such

as the data types like strings or int, where the semantics is the meaning of the data. For instance, when describing an address, the syntax of the city could be a string, but the street and state may also be represented as strings, so some distinction must be made to understand it. Thus, the idea of a street, city, and state needs to be enforced. The information model is critical in services that are outside the boundary of ownership of the consumer where the data being transmitted is completely unknown.

The behavior model represents the actions that can be performed against the service, responses to those actions, as well the process of interacting with it. As an example, a database service may have the actions that can be performed be presenting security authorization, running queries, and updating records. These actions need to be done in some order for an interaction to be successful. Much like the information model, the behavioral model can be decomposed, where the behavioral model is broken into the action model and process model.

The action model represents the possible actions that may be invoked against the service as well as the expected behavior of those actions. Since internal actions are private, this behavior may not be explicitly included in the action model, but its implied effects will be. The process model regards the temporal aspects of how to interact with the service. This, however, is not explicitly defined within the reference model, as it may include aspects that are not strictly part of SOA, but is nonetheless an important aspect that must be defined at least to address the interactions with the service itself. The final step is the real-world effect, or the reason that the consumer is using this service. Consider a flight reservation service. The real-world effect is that of a reserved seat on that specific flight. This effect must be done through some state change in a shared resource as all internal actions that are performed by any participant are private. Through the accumulated changes to this state, a real-world effect is realized. These accumulated changes are closely tied with the execution context of the interaction (to be described), which essentially provides the environment of the interaction.

With the dynamics of the service having been described, the concepts that support them should be addressed for a deeper understanding of all that is SOA. If one recalls, the service description is one such concept that supports the functionality of the service.

The service description lays out what the services capabilities are and is the means to which a consumer can decide if the service is suitable to fulfill its needs. The service description should contain information about the service's existence and reach ability, the set of functions that it provides, the policies and constraints that it runs under and that it will comply, to some extent, with the policies and constraints that consumer imposes, as well as what messages may be exchanged (information model) and how to interact with the service (behavior model). These may not need to

be explicitly included through the use of links or references to other descriptions that include them, such as links to broad policies, in order to take advantage of reusability.

The second concept that supports the dynamics of a service is that of the policies and contracts that constrain the service. Policies represent some constraint on the use, deployment, or description of some owned entity of a participant. One example is that a consumer requires that all data be encrypted. A contract is just an agreement between two or more parties. These concepts are analogous to the policies and contracts that relate to those that are seen in everyday life and can be mapped as such, so no further details will be provided.

The remaining concept is that of an execution context as briefly described. The execution context, more specifically, is the infrastructure elements, policies and constraints, and process of a specific instantiation of a service. This effectively acts as the path between the consumer and provider. As previously stated, it can be thought of as the environment in which the interaction takes place, in that it considers the interaction as a whole and changes as throughout the life of a specific interaction. As such, it is also the context in which the data is interpreted, which also allows for a consumer to distinguish which instantiation of the same service it can use by inspecting the execution context. With this ability, it is then possible to use multiple services as long the execution context can be transmitted within the message that is being exchanged. This idea of statelessness is encouraged, so as to take advantage of the lack of dependency on a single service instantiation, but one must also recognize that there may be cases where state is necessary [35].

These underlying concepts of SOA are critical in understanding what is required at a minimal level for such a solution. The intent of the reference model is just that, to capture the essence of SOA and to provide a common understanding so as to be a base point for the definition of more specific reference architectures or the implementation and design of specific architectures. With this knowledge one can deploy SOA solutions regardless of the current technologies and standards.

4.4. SOA AND OTHER TECHNOLOGIES

Service Oriented Architecture is used because of its very loose coupling between separate objects and data. SOA uses object oriented programming techniques to achieve this loose coupling. Loose coupling means that each part of the program is self-reliant and that minimal assumptions are made regarding the interfaces. A self-reliant process is one that is given data, runs processes on the data, and/or stores the results. It does this without the help of another process. The developer using this process does not care how the process manipulates or stores the data as long as the correct data is given when requested.

The advantage to developing software in this style is that if any underlying algorithms are changed as an update to the object, the interface with the object will not change. The developer will continue to use the process in the same manner regardless of what changes occur in the implementation of the process. This concept allows for a very long lifespan of the process. The lifespan of the systems using this process will also increase.

Another important aspect of an SOA system is that new processes are created with the idea they will be reused in another system later. Any processes not specifically created for a given system have most likely already been designed and fully tested for use from another system. The advantage to this type of design is that it is very easy to implement this type of system. The processes which are being reused have already been fully designed and tested.

These processes just need to be used in the current system. The only parts which need to be designed in this system are the new processes which have not been implemented yet. Overall using the concepts of SOA will decrease the time needed to build the system and increase the time the system can be used without a major overhaul. The downside to the SOA design style is that with the very loose coupling the abstraction becomes more complicated. If two processes, A and B, are loosely coupled, accessing data in Process B from Process A can become an expensive task. Older technologies did not run into problems such as these. Most systems were designed and were very tightly coupled. All of the different parts of the system rely heavily on each other to store, process, and/or retrieve the data. The advantage to systems designed using this methodology is that they are very efficient in running, due to the fact that the data is very accessible at any part in the system. Each part of the system is streamlined specifically for this system. Another advantage to this type of system is that the information is very secure since each system is designed specifically for the type of data being stored. The disadvantages to this type of system is that individual parts of the system are not reusable and if any changes are made, a large part of the system will need to be updated. To implement any new technologies which become available, the system will need to be given a complete overhaul. This fall back will drastically reduce the lifespan of a given system.

ORB is an example of the older technology. An ORB is middle ware software which allows a developer to make program calls from one computer to another via a network. Some ORB uses an **IDL** to describe the data which is being sent across the network. Given a message, the IDL would describe the data in the message so the computer receiving the message would understand what was being received. Encapsulating this was the concept of a **RPC**. This would work using two messages. The first message would request a program to be run using the given parameters and the second message would contain the results of the program.

CORBA is an example of an ORB. CORBA would send messages between systems as part of the RPC protocol. It was best suited for a high volume number of transactions needing security

such as bank or retail transactions. The CORBA systems are very fast because they are very tightly coupled and designed specifically for the purpose it is serving. A disadvantage to CORBA is that it only works with other systems already using CORBA.

These types of systems, using an ORB type of design, are usually fairly complex, just from the nature of the system. However, the same systems created using SOA methods are much more complex. This is due to the fact that the individual processes and parts of the system are too loosely coupled. Extra work is needed to tie these loosely coupled parts of the system together and make the SOA system more complicated. An advantage SOA has over an ORB system is that SOA is able to communicate between two programs each written in any language. The ORB systems do have IDL to help communicate, however an IDL does have to be written to translate one language data into another language. The good part of this is that most languages already have an IDL written for them. The downside is that any new languages will need to have an IDL created specifically for them. SOA already has a structured way to send the data between computers. A message can be sent from any computer to any other computer using any type of language. The receiving computer is able to understand the data based upon the header of the message.

When designing a system, the biggest question to be answered is re-usability vs. performance. If a system is going to need to run fast for a very specific application, then CORBA or an ORB designed system is an appropriate choice. The system is not very easily updated or reused, however it will be much faster than an SOA designed system. If the system is going to be constantly updated and speed is not necessary, then an SOA designed system is very well suited. It will be able to be reused in the future and allow for similar applications to be easily designed off of the original.

The repository architectural style is an example of a cross between SOA and ORB systems. In a repository design, all of the data is centrally stored in relation to the processes involved with the data. Each of the processes using the data are loosely coupled, however the processes with the data are tightly coupled together. The advantage to this is that the data can be stored in a very controlled location and be very secure. Also, new processes can be added into the system very easily since individual processes are loosely coupled. The disadvantage to this type of design is that the repository can become a bottleneck when all of the processes attempt to use the database at once. Because of the very tight coupling between the repository and the processes, if any changes do occur in the repository then changes will also need to take place in the related processes. The repository architectural style is best used for database management systems, such as payroll or bank systems.

The repository style is similar to SOA systems since the processes are very loosely coupled. This allows for new processes to be added very easily. This style is also very similar to the ORB

systems because of the tight coupling of the data to its process. Each individual process is designed with the specific data in mind. This increases the efficiency of each process being run on the data.

SOA is a great new technology however a few things do need to be improved upon. Since many SOA systems are used over the internet between different languages, there is no good way to secure the data in each message. One solution is to use which has already been implemented is through the use of Web Services Security (WS-Security). This type of protocol incorporates security in the header of the message. An example can be found in the HTTPS protocol. Another problem with SOA is that there is no good way to describe unstructured data in the header of a message. Unstructured data may be binary data recognizable only to the receiving computer, however there is no way to tell the receiving computer what exactly that data is.

Compared to its ancestors such as ORB designed services, SOA is almost a completely opposite approach to design. ORB services are designed around very tightly coupled systems, run very efficiently, and need a certain level of security. SOA services are designed with the idea of being easy to create, update, and reuse for similar systems in the future. The repository style sits somewhere in between both the SOA and ORB design styles, containing both positives of each type of design. The best design choice for a given system is for the pattern which will best fit the final system, not the newest concept. In today's world of needing a well built system which will last for a long time and be reusable, an SOA system is an excellent design strategy.

4.5. ARCHITECTURAL MODEL AND TECHNOLOGIES

Influence of business

While IT executives have been facing the challenge of cutting costs and maximizing the utilization of existing technology, at the same time they have to continuously strive to serve customers better, be more competitive, and be more responsive to the business's strategic priorities.

There are two underlying themes behind all of these pressures: Heterogeneity and change. Most enterprises today contain a range of different systems, applications, and architectures of different ages and technologies. Integrating products from multiple vendors and across different platforms were almost always a nightmare. But we also cannot afford to take a single-vendor approach to IT, because application suites and the supporting infrastructure are so inflexible.

Change is the second theme underlying the questions that today's IT executives face. Globalization and e-business are accelerating the pace of change. Globalization leads to fierce competition, which leads to shortening product cycles, as companies look to gain advantage over their competition. Customer needs and requirements change more quickly driven by competitive offerings and wealth of product information available over the Internet. In response the cycle of competitive improvements in products and services further accelerates. Improvements in

technology continue to accelerate, feeding the increased pace of changing customer requirements. Business must rapidly adapt to survive, let alone to succeed in today's dynamic competitive environment, and the IT infrastructure must enable businesses' ability to adapt.

As a result, business organizations are evolving from the vertical, isolated business divisions of the 1980's and earlier, to the horizontal business-process-focused structures of the 1980's and 1990's, towards the new ecosystem business paradigm. Business services now need to be componentized and distributed. There is a focus on the extended supply chain, enabling customer and partner access to business services.

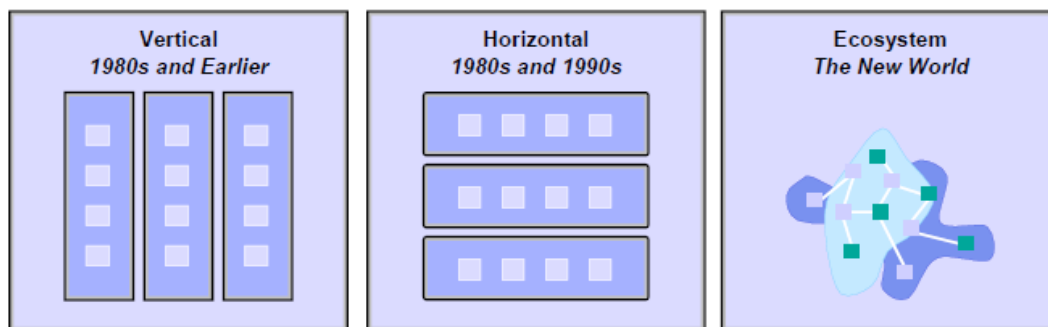


Figure 10: The evolution of business [36]

Object-oriented analysis and design

Larman describes the essence of the object-oriented analysis and design as considering “*a problem domain and logical solution from the perspective of objects (things, concepts, or entities)*” in *Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design*. Jacobson, et al, define these objects as being “characterized by a number of operations and a state that remembers the effects of these operations” in *Object-Oriented Software Engineering: A Use Case Driven Approach*.

In object-oriented analysis, such objects are identified and described in the problem domain, while in object-oriented design; they are transitioned into logical software objects that will ultimately be implemented in an object-oriented programming language.

With object-oriented analysis and design, certain aspects of the object (or group of objects) can be encapsulated to simplify the analysis of complex business scenarios. Certain characteristics of the object(s) can also be abstracted so that only the important or essential aspects are captured, in order to reduce complexity [36].

Component-based design

Component-based design is not a new technology. It is naturally evolved from the object paradigm. In the early days of object-oriented analysis and design, fine-grained objects were marked as a mechanism to provide “reuse”, but those objects are at too low a level of granularity and there are no standard. Coarse-grained components have become more and more a target for reuse in application development and system integration. These coarse-grained components provide certain well defined functionality from a cohesive set of finer-grained objects. In this way, packaged solution suites can also be encapsulated as such “components”. Once the organization achieves a higher level of architectural maturity based on distinctly separate functional components, the applications that support the business can be partitioned into a set of increasingly larger grained components.

Components can be seen as the mechanism to package, manage and expose services. They can use a set of technologies in concert: Large-grained enterprise components, that implement business-level use-cases, can be implemented using newer object-oriented software development in combination with legacy systems [36].

Service-oriented design

In *Component-Based Development for Enterprise Systems*, Allen includes the notion of services, describing a component as “an executable unit of code that provides physical black-box of encapsulation of related services. Its service can only be accessed through a consistent, published interface that includes an interaction standard. A component must be capable of being connected to other components (through a communications interface) to a larger group”.

A service is generally implemented as a coarse-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled, message-based communication model. **Figure 11** shows important service-oriented terminology:

- **Services:** Logical entities, the contracts defined by one or more published interfaces.
- **Service provider:** The software entity that implements a service specification.
- **Service consumer (or requestor):** The software entity that calls a service provider. Traditionally, this is termed a “client”. A service consumer can be an end-user application or another service.
- **Service locator:** A specific kind of service provider that acts as a registry and allows for the lookup of service provider interfaces and service locations.
- **Service broker:** A specific kind of service provider that can pass on service requests to one or more additional service providers.

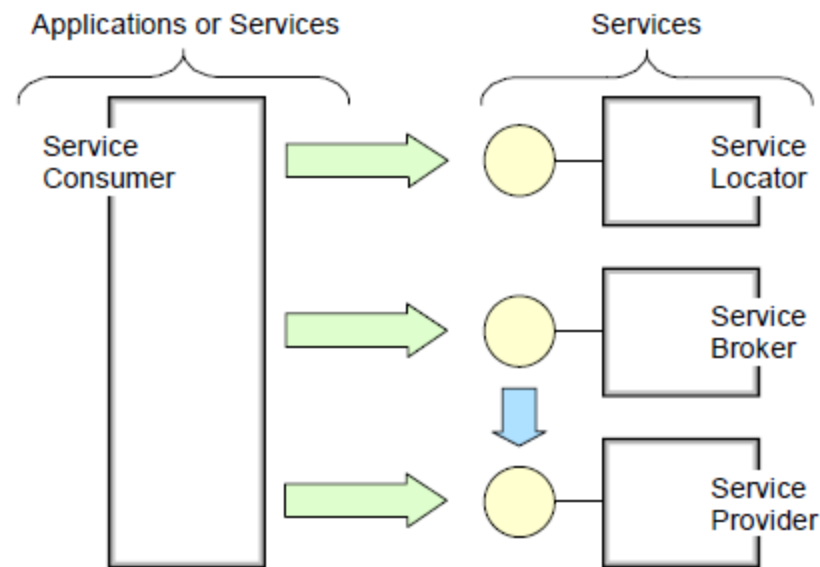


Figure 11: Service oriented terminology [36]

Interface-based design

In both component and service development, the design of the interfaces is done such that a software entity implements and exposes a key part of its definition. Therefore, the notion and concept of “interface” is a key to successful design in both component-based and service-oriented systems. The following are some key interface-related definitions:

- **Interface:** Defines a set of public method signatures, logically grouped but providing no implementation. An interface defines a contract between the requestor and provider of a service. Any implementation of an interface must provide all methods.
- **Published interface:** An interface that is uniquely identifiable and made available through a registry for clients to dynamically discover.
- **Public interface:** An interface that is available for clients to use but is not published, thus requiring static knowledge on the part of the client.
- **Dual interface:** Frequently interfaces are developed as pairs such that one interface depends on another; for example, a client must implement an interface to call a requestor because the client interface provides some callback mechanism.

Figure 12 shows the UML definition of a **CRM** service, represented as a UML component that implements the interfaces **AccountManagement**, **ContactManagement**, and **SystemsManagement**. Only the first two of these are published interfaces, although the latter is a public interface. Note that the **SystemsManagement** interface and **ManagementService** interface form a dual interface. The CRM service can implement any number of such interfaces, and it is this ability of a service (or component) to behave in multiple ways depending on the client that allows

for great flexibility in the implementation of behavior. It is even possible to provide different or additional services to specific classes of clients. In some run-time environments such a capability is also used to support different versions of the same interface on a single component or service [36].

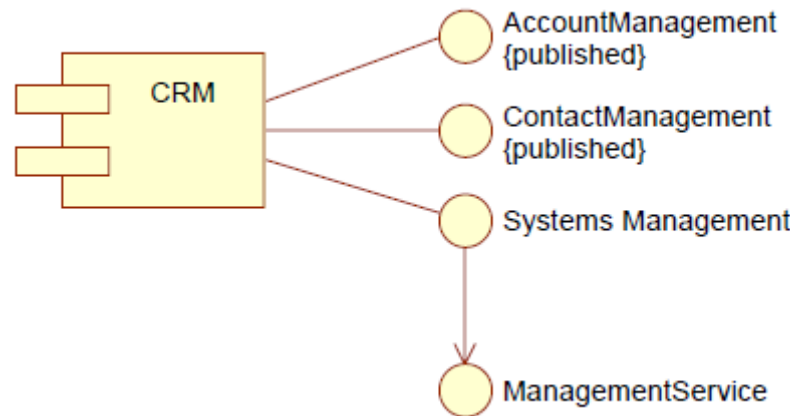


Figure 12: Implemented services [36]

WS security

The main goal of SOA security standards is to provide a basis for interoperability among multiple products used in heterogeneous customer environments. Standards-based implementation strategies accelerate development, simplify integration, and reduce administrative costs over time. Most SOA industry standards are defined in XML frameworks. The last few years have seen the emergence of a plethora of XML-based specifications addressing various aspects of SOA security.

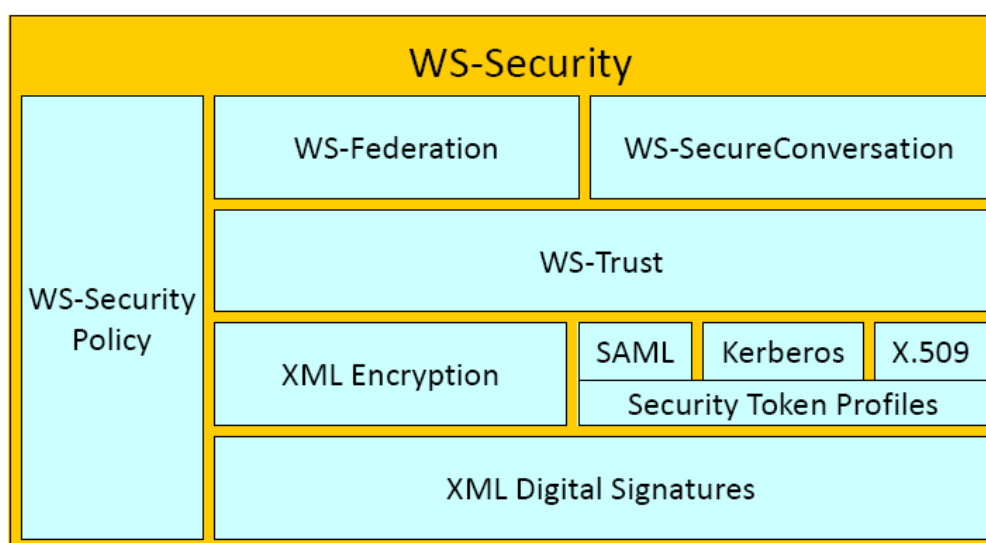


Figure 13: WS-Security technology umbrella [37].

Most of these specifications are part of the so-called WS-* (Web Services specifications) stack. Most WS-* specifications started as proprietary industry initiatives. Some of these specifications (e.g., WS-Security, WS-Trust and WS-Policy) have been transferred over to standards bodies such as the Organization for the Advancement of Structured Information Standards (OASIS) or the World Wide Web Consortium (W3C). WS-* specifications often depend on each other, for example, WS-Policy uses WSPolicyAssertions. WS-* specifications also leverage non-WS specifications, for example, WS-Security uses XML Encryption and XML Signature [38].

WS-Security specifies SOAP security extensions that provide confidentiality using XML Encryption and data integrity using XML Signature [39]. WS-Security also includes profiles that specify how to insert different types of binary and XML security tokens in WS-Security headers for authentication and authorization purposes:

- Username with optional Password digest (defines how a web service consumer can supply a username as a credential for authentication; the username can be accompanied by a hashed password).
- X.509 Certificate (a signed data structure designed to send a public key to a receiving party).
- Kerberos ticket (an authentication and session token).
- Security Assertion Markup Language (SAML) assertion (see more detail on SAML later in this document).
- REL document (rights expression language (REL) license tokens inserted in WS-Security headers are used for authorization).
- XCBF document (defines how to use the XML Common Biometric Format (XCBF) language for authentication with the WS-Security specification).

ESB

To integrate old and new, service-oriented architecture (SOA) needs an infrastructure that can connect any IT resource, whatever its technology or wherever it is deployed. To be flexible, it needs an infrastructure that can easily combine and re-assemble services to meet changing requirements without disruption. And to be dependable, it needs an infrastructure that is robust and secure.

An ESB is software infrastructure that simplifies the integration and flexible reuse of business components within a service-oriented architecture. An ESB provides a dependable and scalable infrastructure that connects disparate applications and IT resources, mediates their incompatibilities, orchestrates their interactions, and makes them broadly available as services for additional uses.

In most organizations, technological heterogeneity is more the rule than the exception. To integrate applications within an SOA, it is necessary to both span new service-enabled applications as well as existing applications. An enterprise service bus simplifies connection of new applications, web services, and hundreds of other technologies, including batch files, application servers, legacy middleware products and packaged applications.

Built-in mediation capabilities make it straightforward for the ESB to reconcile the incompatible protocols, data formats and interaction patterns of disparate connected resources. Data represented in XML is readily transformed using XSLT or XQuery services and routed to particular downstream systems based on their content or other attributes. Data may be split, aggregated, and enriched en-route to a consuming service. For example, a batch file containing customer orders can be broken down into individual transactions, with each transaction enriched with customer and product information before subsequent routing and delivery to specialized order fulfillment systems. Order fulfillment systems best served by batch files can be incorporated by aggregation of the transaction flow back into batch form. Intermediary services such as these allow integration architecture to eliminate dependencies between service consumers and service producers, making it easier to create a loosely-coupled system that can be changed without disruption.

Service orchestration provides a means to automate the invocation of multiple services, managing complex flow logic and process state, and correlating responses from downstream systems to a given orchestration instance. Orchestration supports process automation by invoking services in a particular order and by following a set of rules. Orchestration offers a top-down, process-oriented approach to SOA, allowing the modeling of a process in the abstract, followed by subsequent specification of technical details for the implementation of individual process steps. Service orchestration also makes it easier to monitor and manage the execution of these processes as they run, and also makes it easier to change process definitions to adapt to new business requirements.

Finally, an enterprise service bus makes services available for broad access and re-use. Here, communications capabilities of the underlying enterprise messaging backbone come to the fore. Properly engineered, they provide not just the dependable, robust and secure communication for service access, but also support communication across firewall's and LAN boundaries in a seamless manner. The enterprise messaging backbone also provides the publish-and-subscribe semantics that support an event-driven model in which any number of service consumers can respond to a business event. These rich messaging semantics enhance service re-use by make it easy to dynamically add new service producers and consumers to the scenario at runtime, without requiring a recoding of a process or service.

Users of an enterprise service bus typically follow an incremental, leave-and-layer approach to project implementation. Existing applications and SOA infrastructure are left in-place. Whether automating or streamlining a process, gathering and disseminating information, or responding to business events, the ESB is deployed and configured to accomplish a particular business change objective. At the same time, the ESB lays a highly flexible and scalable architectural foundation for additional projects, providing the basis for extension and re-use of these projects to meet additional user requirements. The incremental approach allows the rapid delivery of value to users, thereby reducing the risk that requirements change before the project is complete, and at the same time executing on the vision of an SOA for long-term business agility and broad-scale interoperability [40].

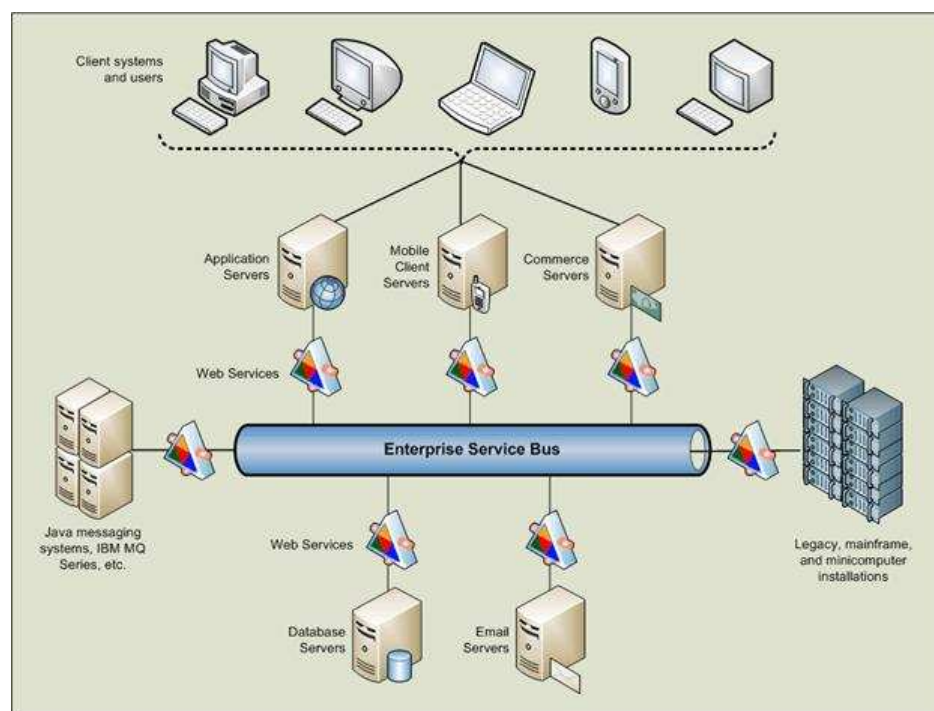


Figure 14: Enterprise Service Bus model [41]

Because ESB architecture and function varies from vendor to vendor, when selecting ESB technology for integration within an SOA, it is necessary to inspect a given implementation closely. There are five principal aspects that need to be considered when selecting an ESB.

Performance and scalability

It is essential that the infrastructure used to integrate within an SOA provide the performance expected of enterprise systems while easily accommodating changes in demand. It is critical that an ESB not create bottlenecks or chokepoints in the infrastructure that would restrict the

breadth of its deployment or limit the throughput of data it can carry to and from connected resources. Unlike traditional hub-and-spoke integration broker technologies, an enterprise service bus with a distributed architecture will facilitate the distributed processing of services, service orchestration and the underlying communication infrastructure. While distribution is an effective means to ensure high throughput and scalability, it is important that physical deployment not impact the logical or semantic behavior of the infrastructure, and that a highly distributed infrastructure be straightforward to manage, monitor and control.

Security, reliability and availability

An ESB should provide configurable enterprise-level quality of service to ensure that service communication is as secure and reliable as needed to meet a particular business requirement. By providing these guarantees, service developers can depend on these features, dramatically simplifying the job of service development and focusing activity on development of business logic, not integration logic. For example, producer services themselves need not manage data re-transmission if consumer services are temporarily unavailable; infrastructure which guarantees delivery will store the data and forward it when the consuming service is available. Highly available SOA integration infrastructure isolates applications from faults resulting from server and communication failures. A true highly available ESB will present itself to services as not only a reliable infrastructure, but one whose integration services are continuously available, even if a component in its distributed infrastructure should fail.

Distribution

In an SOA, services and service orchestrations will interact with services spread across an organization, and between organizations. An ESB provides the communications facilities which link them together with configurable qualities of service and messaging semantics, with both queuing and publish-and-subscribe behavior. This allows the gathering and distribution of data, as well as the notification of business events and correlation of replies of subscribing services. In a federated SOA environment—one in which services cross organizational boundaries – or in a distributed environment—one in which service communications bridge geographic boundaries – it is critical that data, events, and replies are directed to the right place at the right time, without management overhead. Advanced distribution capabilities allow flexible extension of the ESB to incorporate and connect resources in additional security domains without any reconfiguration or disruption of running systems.

Flexibility

The flexibility of a properly architected enterprise service bus allows an organization to change orchestration, rules, data mapping and relationships between applications with minimal effort and disruption. Here, the core requirement is that it be possible to dynamically change services, process and schema without disruption of running systems. At the same time, the architecture must afford complete freedom to define and change physical deployment for efficient, scalable processing without any impact to the semantics or behavior of integrations. Sophisticated publish-and-subscribe semantics allow the simultaneous broadcast of information or notifications of business events to a potentially large number of interested parties, and also make it easy to expand the system without making changes to the service provider. Service mediation eliminates inflexible, hard-coded service interdependencies, which are the enemy of SOA flexibility because they make it difficult to understand the impact of a change in a service. By isolating services from one another, the ESB creates a framework for loosely-coupled integration, thereby enabling much easier management of change in a complex environment. Finally, service orchestration provides a flexible way to model, define, and manage processes that direct the sequenced interaction of services. Advanced service orchestration facilities will provide the capability to manage both high-throughput and long-running process scenarios in the ESB.

Visibility and control

An ESB should manage and monitor the infrastructure as well as the processes and services deployed within it. The first task of management is the deployment of services, which may be hosted anywhere the ESB runs. Since a deployment may be highly distributed, such as a retail deployment comprising hundreds or thousands of stores, the ESB should make it easy to deploy and upgrade services remotely from a central location. To provide dynamic, run-time control, an enterprise service bus is configuration-driven: services are parameterized and their mediated relationships are declared, so they may be changed without re-compilation and re-deployment. This allows the modification of data and process flow without shutting down running services. For federated environments, it is critical that the ESB management domain seamlessly bridge LAN boundaries and security domains; the ESB should allow visibility and control across the entire federated environment. To facilitate development in a real-world, distributed environment, the ESB needs real-time message tracking and distributed flow control debugging. Finally, in order allow the diagnosis and management of problems in complex distributed systems, administrative features should provide logging and auditing of services, as well as the monitoring of faults, service and process status, and detailed performance statistics of the ESB communication infrastructure.

Web services

A web service is a piece of business logic, located somewhere on the Internet, that is accessible through standard-based Internet protocols such as HTTP or SMTP. Using a web service could be as simple as logging into a site or as complex as facilitating a multi organization business negotiation. Given this definition, several technologies used in recent years could have been classified as web service technology, but were not. These technologies include win32 technologies, J2EE, CORBA, and CGI scripting. The major difference between these technologies and the new breed of technology that are labeled as web services is their standardization. This new breed of technology is based on standardized XML (as opposed to a proprietary binary standard) and supported globally by most major technology firms. XML provides a language-neutral way for representing data, and the global corporate support ensures that every major new software technology will have a web services strategy within the next couple years. When combined, the software integration and interoperability possibilities for software programs leveraging the web services model are staggering [42].

A web service has special behavioral characteristics:

- ***XML-based:*** By using XML as the data representation layer for all web services protocols and technologies that are created, these technologies can be interoperable at their core level. As a data transport, XML eliminates any networking, operating system, or platform binding that a protocol has.
- ***Loosely coupled:*** A consumer of a web service is not tied to that web service directly; the web service interface can change over time without compromising the client's ability to interact with the service. A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must also be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.
- ***Coarse-grained:*** Object-oriented technologies such as Java expose their services through individual methods. An individual method is too fine an operation to provide any useful capability at a corporate level. Building a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service. Businesses and the interfaces that they expose should be coarse-grained. Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic [42].
- ***Ability to be synchronous or asynchronous:*** Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing. Asynchronous

operations allow a client to invoke a service and then execute other functions. Asynchronous clients retrieve their result at a later point in time, while synchronous clients receive their result when the service has completed. Asynchronous capability is a key factor in enabling loosely coupled systems.

- **Supports RPC:** Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support [42]. Component development through Enterprise JavaBeans (EJBs) and .NET Components has increasingly become a part of architectures and enterprise deployments over the past couple of years. Both technologies are distributed and accessible through a variety of RPC mechanisms. A web service supports RPC by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component.
- **Supports document exchange:** One of the key advantages of XML is its generic way of representing not only data, but also complex documents. These documents can be simple, such as when representing a current address, or they can be complex, representing an entire book or RFQ. Web services support the transparent exchange of documents to facilitate business integration.

The Major Web Services Technologies

Several technologies have been introduced under the web service rubric and many more will be introduced in coming years. In fact, the web service paradigm has grown so quickly that several competing technologies are attempting to provide the same capability. However, the web service vision of seamless worldwide business integration is not being feasible unless the core technologies are supported by every major software company in the world [42].

Over the past two years, three primary technologies have emerged as worldwide standards that make up the core of today's web services technology. These technologies are:

- **Simple Object Access Protocol:** SOAP provides a standard packaging structure for transporting XML documents over a variety of standard Internet technologies, including SMTP, HTTP, and FTP. It also defines encoding and binding standards for encoding non-XML RPC invocations in XML for transport. SOAP provides a simple structure for doing RPC: document exchange. By having a standard transport mechanism, heterogeneous clients and servers can suddenly become interoperable. .NET clients can invoke EJB exposed through SOAP, and Java clients can invoke .NET Components exposed through SOAP.

- **Web Service Description Language:** WSDL is an XML technology that describes the interface of a web service in a standardized way. WSDL standardizes how a web service represents the input and output parameters of an invocation externally, the function's structure, the nature of the invocation (in only, in/out, etc.), and the service's protocol binding [42]. WSDL allows disparate clients to automatically understand how to interact with a web service.
- **Universal Description, Discovery, and Integration:** UDDI provides a worldwide registry of web services for advertisement, discovery, and integration purposes. Business analysts and technologists use UDDI to discover available web services by searching for names, identifiers, categories, or the specifications implemented by the web service. UDDI provides a structure for representing businesses, business relationships, web services, specification metadata, and web service access points.

BPEL

Business Process Execution Language (BPEL) is a XML-based language used to define enterprise business processes within Web services. Every company has its unique way of defining its business process flow. The key objective of BPEL is to standardize the format of business process flow definition so companies can work together seamlessly using Web services. BPEL extends the Web services interaction model and enables it to support business transactions. BPEL is based on Web services in the sense that each of the business process involved is assumed to be implemented as a Web service. Processes written in BPEL can orchestrate interactions between Web services using XML documents in a standardized manner. These processes can be executed on any platform or product that complies with the BPEL specification.

BPEL supports two different types of business processes:

- **Executable processes:** Models the actual behavior of a participant in a business interaction. They follow the orchestration paradigm and can be executed by an orchestration engine.
- **Abstract processes:** Uses process descriptions that specify the mutually visible message exchange behavior of each of the parties involved in the protocol, without revealing their internal behavior.

BPEL is a convergence of language features from IBM's Web Service Flow Language (WSFL) that uses a directed graph approach and Microsoft's XLANG, which is the orchestration language, used in BizTalk server and has a block-structured approach. Both WSFL and XLANG are now superseded by the BPEL specification.

BPEL 1.0 was jointly developed by IBM, BEA, SAP, Siebel, and Microsoft in August 2002. In April 2003, BPEL 1.1 was submitted to OASIS to obtain even broader industry acceptance and open standardization.

BPEL is based upon Web services and hence is related to standards such as WSDL, XML, SOAP, and UDDI. The following diagram describes the relations between the various standards within the Web services technology stack [43].

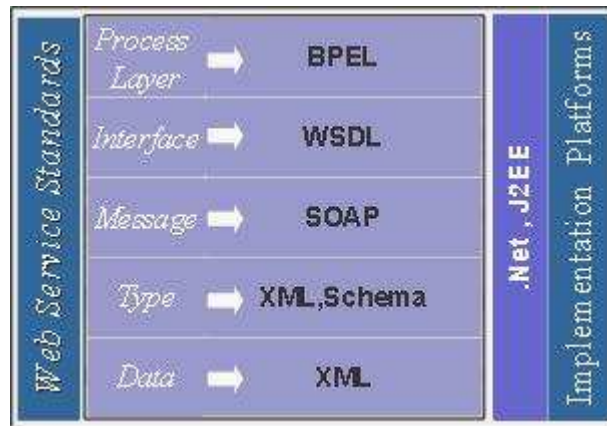


Figure 15: BPEL relation to other web service standards [43].

4.6. INTEROPERABILITY AND TRUSTED WEB SERVICES

Web Service implementations support interoperable machine-to-machine interaction over a network. Applications that are written in different programming languages and running on various platforms can use Web Services to exchange data over a computer network, due to its inherent support for interoperability. The key factor to achieving interoperability is the use of open standards (such as WSDL, SOAP); thus, Web Services are becoming the foundation of SOA.

SOA expresses a new software architecture, which uses services to fulfill the user requirements. Recently, SOA has gained a broader industry acceptance as Web Service standards became more popular. However, some obstacles still remain. One of the major concerns is the security of Web Services. It is a valid and reasonable concern as the practice has been that security issues were not carefully studied and integrated in the design stage of Web Service standards. The existing “Web Service security solutions” are rather add-on modules to the already-defined Web Service standards.

The existing Web Service security solutions authenticate Web Service requests based on the information provided by clients. Information supplied by clients varies depending on particular protocols being used. Examples of such information are username/password combination, web certificate, security token, bio-metric reading, and so on. However, these solutions have one common drawback. Since the information is provided by the Web Service requester, it is hard for

the service provider to distinguish the information coming from the “right” source vs. a malicious source. The Web Service provider has very limited capabilities to identify and isolate the service requests coming from a compromised client.

While it is important to remotely measure and verify the integrity of hardware and software components of the Web Service requester, their solution would be insufficient in two points. First, some of their work is overlapping with existing **TCG** standards, or more specifically, Trusted Network Connect (TNC) standards. As TNC standards have been designed to be interoperable and comparable with existing solutions, they are more likely to be adopted by companies and vendors. Second, in their WS-attestation proposal, all attestation validation happens at the Web Service layer, which inevitably degrades the performance of the Web Service and eventually leads to scalability issues.

4.7. PRODUCTS FOR SOA BASED SYSTEMS

Oracle SOA Suite

Oracle has spent several years acquiring technology, and just as long putting the pieces together into a cohesive offering – including Collaxa for BPEL, Oblix for WS-Security and WS-Management, and branded Systinet BSR 6.0 for its registry/repository. Toss in a rules engine and the result is SOA Suite, a nearly seamlessly integrated set of tools designed to enable SOA implementations in the enterprise.

The primary component in SOA Suite is Oracle BPEL PM 10.1.2. SOA Suite is a J2EE offering that can be deployed in a number of containers, including IBM's WebSphere, BEA Systems WebLogic, JBoss and OracleAS. By default, the suite installs in an OracleAS container with an embedded OracleLite meta-data repository.

SOA Suite differs from most of the other products. Instances of SOA Suite do not communicate, so it couldn't be designated backup nodes as it could with Fiorano product, but because all state information is stored in a shared repository, instances should be able to re-create themselves, regardless of which node they were instantiated on. Failover requires that load balancing be configured at the container level, or that you use an external, hardware load balancer such as those from F5 Networks or Citrix. Failover using BEA AquaLogic is accomplished using WebLogic Server clustering technology, while Cape Clear relies on the clustering capabilities of the J2EE enterprise service platform in which it is deployed, with some routing-based failover available at the ESB layer provided the container handling the routing isn't the one that fails, of course.

Like TIBCO product, Oracle allows for profiling individual orchestrations, providing execution times on a per-activity basis. But unlike TIBCO, Oracle presentation of the data in SOA Suite Web-based console was easy to navigate. Unique to SOA Suite is the ability to fully manage

in-flight orchestrations and execute multiple versions of the same orchestration. Still, SOA Suite lacks AquaLogic granularity of monitoring in this regard, and exposing process variable data through AquaLogic monitoring capabilities is far easier than accomplishing the same task with Oracle SOA Suite.

HTTP2HTTP and JMS2JMS services in the same container, for example, can communicate without requiring the overhead of setting up the underlying TCP session for an HTTP exchange. This feature was offered by BEA suite as well--except for SOAP/HTTP services, which always required us to set up the underlying TCP session.

SOA Suite's design-time environment is available as a JDeveloper or Eclipse plug-in. Oracle uses the same sort of environment for building transformations that Fiorano employs, but Oracle is a bit easier to use which is somewhat surprising as many of Oracle tools are highly developer-oriented. Although both are graphical and use a drag-and-drop mechanism for building up expressions, Oracle presents the various functions, like “*concat*”, without exposing you to developer-oriented parameterization requirements.

SOA Suite's default XSLT engine is Xalan, with Xerces as its default XML parser, but both use a pluggable model and can be replaced with another parser and engine if so desired [45].

WebSphere Business Integration Server Foundation

IBM recently announced the availability of WebSphere Business Integration Server Foundation V5.1. This product is one of several business integration offerings provided by IBM, each designed to address a different set of customer and application needs.

WebSphere Business Integration Server Foundation V5.1 represents IBM approach to building and deploying SOA-based applications that can adapt quickly and easily to change. It is designed to support the creation of reusable services (either new ones or those based on existing services, back-end systems, Java assets, and packaged applications). Services can then be combined to form both composite applications and business processes, which can further leverage business rules to make these applications and business processes “adaptable” [46].

Over the past few years, IBM has positioned the WebSphere Application Server and its various versions and implementations as the foundation for moving forward with SOA-based implementations. Today, while IBM application server offerings still represent an important core element of how it implements an SOA, the company has broadened its SOA strategy to deliver what it calls the Business Integration Reference Architecture (see **Figure 16**).

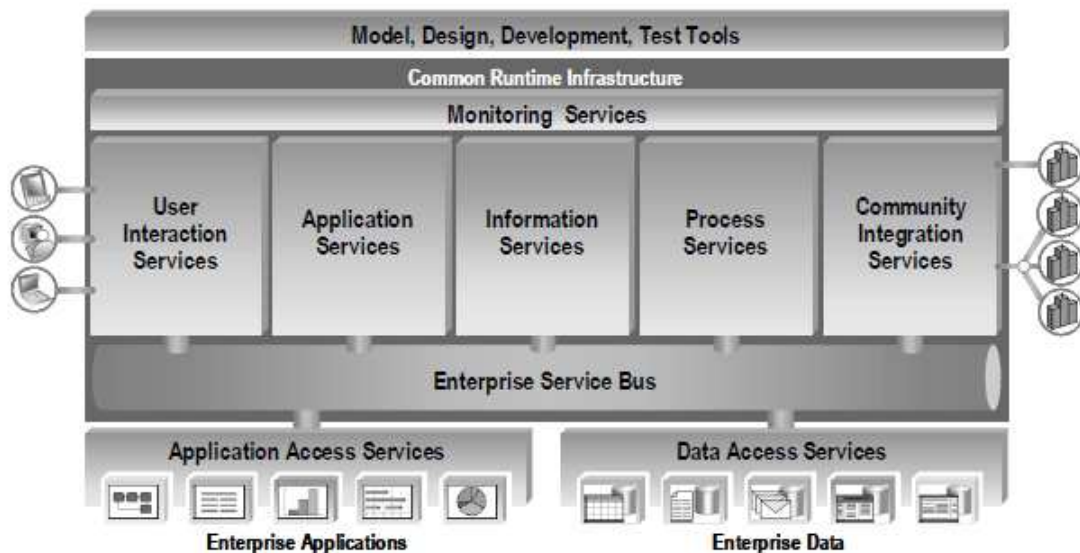


Figure 16: IBM Business Integration Reference Architecture [46].

Instead of viewing services-based implementations as a complex, non-integrated stack of software elements and applications, the architecture instead positions functions as services that are connected via an Enterprise Service Bus. According to IBM, the primary services in this architecture are:

- **Information Services**, which provide the capabilities, required to federate, replicate, and transform data sources.
- **Process Services**, which deliver the control services required to manage the flow and interactions of multiple services in a way that implements a business process.
- **Application Services**, which offer the set of runtime services required for new applications and new business logic.
- **User Interaction Services**, which include the capabilities required to deliver IT functions and data to end users.
- **Community Integration Services**, which provide the document, protocol, and partner management services required for efficient implementation of business-to-business interactions.

WebSphere Business Integration Server Foundation V5.1 represents one of the lead IBM product offerings in the Process Services space.

SAP NetWeaver

SAP NetWeaver is integrated technology platform and is the technical foundation for all SAP applications since the SAP Business Suite. SAP NetWeaver is marketed as a service-oriented

application and integration platform. SAP NetWeaver provides the development and runtime environment for SAP applications and can be used for custom development and integration with other applications and systems. SAP NetWeaver is built using open standards and industry de facto standards and can be extended with, and interoperate with, technologies such as Microsoft .NET, Sun Java EE, and IBM WebSphere [47].

NetWeaver release is considered as a strategic move by SAP for driving enterprises to run their business on a single, integrated platform that includes both applications and technology. Industry analysts refer to this type of integrated platform offering as an “applistructure” (applications + infrastructure). It is widely held that this approach is driven by industry's need to lower IT costs through an enterprise architecture that is at once:

- More flexible;
- Better integrated with applications;
- Built on open standards to ensure future interoperability and broad integration;
- Provided by a vendor that is financially viable for the long term [47].

Microsoft BizTalk Server

Information technology does improve speed, accessibility and timeliness of the already existing information, but, in order to create new information and use the existing information in newer and more innovative ways there is a need for the appropriate tools that are able to evaluate information in novel ways. BizTalk Server is one such tool. Calling it a mere tool would be an understatement! In this article we will explain this new software keeping in mind the benefits that a small and medium enterprise (SME) can gain by deploying it effectively [48].

E-business implementers, Web Service Developers, business analysts, technology planners, project managers, application designers and executives would find BizTalk Server workable software. It empowers you to use the infrastructure provided and build successful e-commerce communities.

To fully appreciate what this next generation software is all about, let us mander for a moment and discuss the powerhouse of markup languages - XML. Exchanging information and data over the Internet can become complex and burdensome, more so, when applications from different companies are written in special unique formats that are difficult to exchange. A lack of a single “technical vocabulary” for describing business data and processes can result in fragile interconnectivity thereby motivating the need for a meta-language like XML that specifies a common code/meaning of all the data that can be exchanged world wide.

Organizations, applications and individuals can communicate far more effectively if they agree on the structure and meaning of information. XML was specifically designed to facilitate such

communication and BizTalk Server provides a solid foundation necessary to wield XML as a strategic tool. Now, before you draw your own conclusions, let me clarify that you don't need to understand XML in order to implement BizTalk Server. A simple knowledge of HTML and relational databases, and a little effort in understanding the product enables easy data integration, workflow and knowledge management.

Microsoft built BizTalk Server as a part of their enterprise application integration in the .NET server family (www.microsoft.com) that constitutes a suite of tools and services like management, administration, mapping and tracking [48].

It includes the following:

- ***BizTalk Server Orchestration Designer:*** A design tool that enables business analysts, developers and IT professionals to interact on a common platform.
- ***BizTalk Editor:*** Creates and edits XML documents.
- ***BizTalk Mapper:*** Transforms one XML document into another XML schema.(Schemas are different ways that an XML document can be structured and predefined in order to simplify data exchange).
- ***BizTalk Messaging Manger:*** Allows the exchange of information and applications with minimal programming.
- ***BizTalk Framework:*** Enables routing, tracking and analyzing sent and received data.
- ***BizTalk Administration Tool:*** Allows end-to-end integration and easy online analytical processing of user-specific information.
- ***Pipeline Editor:*** Tracks document processing from start to finish.

BizTalk Server supports standard Internet Technologies like EDI (Electronic Data Interaction), multiple transports and protocols (HTTP, SMTP), custom and flat file formats.

With the BizTalk Server you can do the following:

- Simplify data exchange - without editing or understanding XML you can create impressive documents and modify them using any text editor.
- Document translation, filtering and routing based on content.
- Conduct on-line transactions easily and cheaply along with the delivery of sound and video.
- Integration of typewritten user interface on Web browsers and backend databases thereby orchestrating purchasing and order processing systems.
- Create applications that can mediate between two or more heterogeneous databases and intelligently apply information that is tailor made to individual users.

- Point by point tracking of data thereupon establishing reliability among clients.
- Establish secure communications that support public key encryption, digital signatures and encryption along with multipurpose Internet Mail Extensions (S/MIME).
- Seamless links with customers, suppliers and partners.
- Integrate wholly unrelated functions like human resources and finances [48].

4.8. CONCLUSION

One of the reasons that SOA has come into prevalence is because it promotes high reusability and loose coupling of services. Services can be thought of as discrete modules that perform well defined operations on their inputs to produce the output. When developing the services to be used in an SOA, engineers focus on creating highly reusable, loosely coupled services that are very good at accomplishing a small number of tasks. However, services in this state are hardly ready to be used directly by consumers. For instance, imagine a credit card processing service that is used by an online retailer. The credit card processor is a specialized service that takes in credit card information and performs the necessary operations on that information (e.g., verification, charging the card, etc.). As a person shopping in the online store, you would never expect to have to deal with the credit card processor directly. Instead, you are presented with an interface that, behind the scenes, orchestrates many of these specialized services into something that is useful (e.g., purchasing an item). In SOA jargon, this concept is known as a business process. In researching what SOA is all about, the reader will see terms like “business processes” and “business rules”. Business processes are simply a figure of speech that indicates the organization of loosely coupled, highly reusable, small services into a set that together can perform a meaningful task. Thus, a business process is simply a particular organization of services that meet an application-specific need and has little to do with how business or companies operate.

Also, the rise of SOA to buzzword status in the software engineering community is closely linked to the web, and in particular web services. In fact, while SOA does not require the existence of web services as we know them today, it is hard to imagine SOA developing as it did without the web and web services. The reason is because web services are a perfect example of the realization of SOA. The past few years have witnessed some interesting developments in web services. Recently a small private company in California made the winning bid, \$15.5 million, for 39 different patents concerning web services. Companies like Google, Amazon, and eBay are also providing inexpensive web services to the community. For example, Amazon’s **SQS** is a web service that distributed applications can use for intra-communication. The Amazon SQS has a simple **API**, and can be accessed by any program much like any other queue is accessed. The advantage here is that, using web services, an application that might otherwise be tightly coupled to

perform communication is now loosely coupled; each part of the application need only know the location of the queue on the web.

Oracle most comprehensive SOA offering, Oracle SOA Suite, is an integrated, best-in-class suite of products that helps you build, deploy, and manage deployments ranging from department-level to enterprise-wide systems. This 100 percent standards-based, hot-pluggable infrastructure interoperates with your existing IT investments. Many Oracle customers have succeeded in deploying high-volume, mission-critical SOA systems by leveraging the industry's best Oracle Service Bus combined with Oracle's renowned extreme-scalability on a grid computing infrastructure.

Oracle based service oriented architecture is a new technology which has been introduced in the past few years. Still in its infancy stages, much work needs to be done before it is perfected. Since reusability is a key factor in getting systems put together quickly, SOA is an excellent choice. The future is unforeseeable, but with the use of SOA while designing, systems will be much more reliable and have a longer lifespan.

5. EDAS – Online Representation of State Social Insurance Board under the Ministry of Social Security and Labour

5.1. INTRODUCTION

In this chapter the practical part of master's work will be described – Online representation of State Social Insurance Board under the Ministry of Social Security and Labour (EDAS). This system based on SOA technological solution and Oracle Middleware based applications.

This chapter briefly describes interoperability level of the systems and variety of systems that are involved in this process.

Due to security credentials system described in high level of abstraction.

5.2. SYSTEM OVERVIEW

Created system allows employers to submit and Information system of State Social Insurance Board under the Ministry of Social Security and Labour to receive electronic notification of service of state social contribution that will be legal to acknowledge and process according received data. Additional information considering employers (or insurer) financial responsibilities, number of insured, etc. will be presented in several most popular formats as PDF, XLS, RTF and HTML.

Latest statistical researches based on analysis of system's logs state that popularity of this system is increasing and expected number of users is surpassed faster that it was determined. EDAS wide spread among employers allows Government to increase its productivity and - what is most important – to enlarge number of online represented governmental institutions. Role of online Government in everyday life will be more and more important. These changes will make strong push to new technologies in Lithuania.

Automation of process of interaction between Government and employer decreases time of notification processing from 8-10 work hours to 1-2 (depending on notification size).

Among mentioned benefits important is that EDAS is based on Internet technologies - this will increase spread of Internet and its usage not only among legal persons but also physical persons. Usage of electronic signature will prepare legal and social bases for proper behavior with such data type.

Integration of EDAS in everyday life and into workflow of State Social Insurance Board will increase productivity and GNP growth in Lithuania.

5.3. ARCHITECTURE

EDAS is created using portal paradigm – all external and/or internal modules can be easily manipulated, provided to other external systems using WSRP specification.

The WSRP specification defines a web service interface for interacting with presentation-oriented web services. Initial work was produced through the joint efforts of the Web Services for Interactive Applications (WSIA) and Web Services for Remote Portlets (WSRP) OASIS Technical Committees. With the approval of WSRP v1 as an OASIS standard in September, 2003, these two technical committees merged and continued the work as the Web Services for Remote Portlets (WSRP) OASIS Technical Committee.

Scenarios that motivate WSRP functionality include:

- Content hosts, such as portal servers, providing portlets as presentation-oriented web services, that can be used by aggregation engines;
- Content aggregators, such as portal servers, consuming presentation-oriented web services, provided by portal or non-portal content providers and integrating them into a portal framework [49].

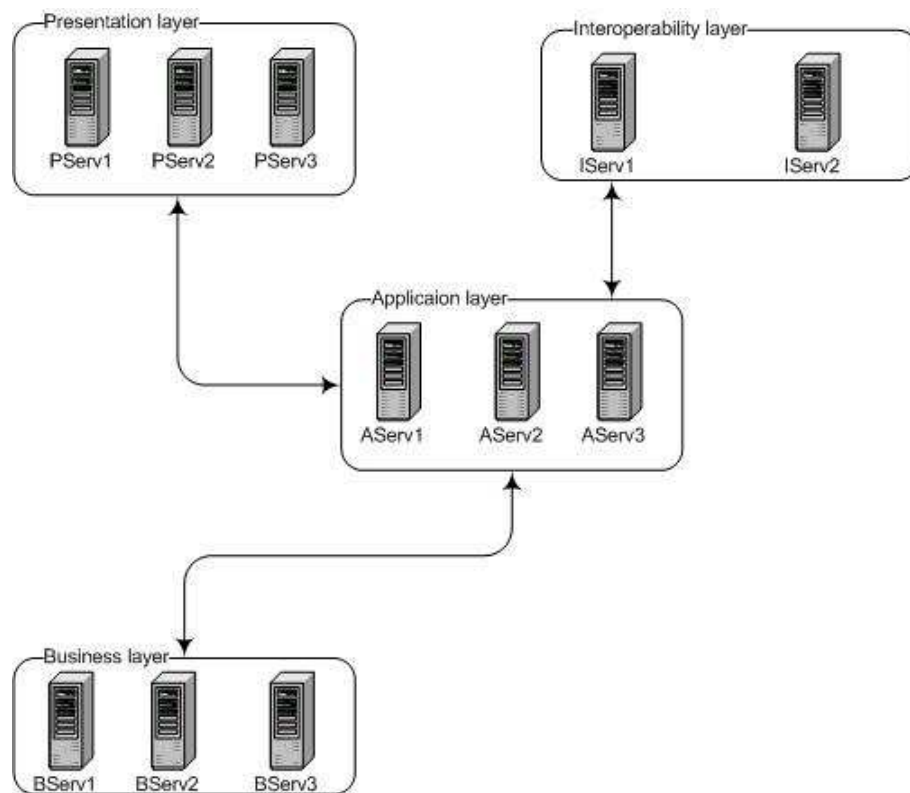


Figure 17: Technical specification of the system.

This approach provides high availability architecture for portal - based solution, which allows scalability of each layer of a system. Architecture has three application layers:

- **Application layer:** holds applications that are generally based on J2EE. These applications were developed and deployed during the phase of trial exploitation. Described applications use the following technologies:
 - o JSR specification based portlets
 - o JSP specification based servlets and JSP.
 - o Struts framework
 - o JDBC
 - o Bouncy Castle provider.
- **Business layer:** contains all business logics and is the critical for the system stability and functionality. The following technologies are used:
 - o PL/SQL programming language
 - o XPath XML manipulation language
- **Interoperability layer:** is the essential layer for the State Social Insurance Board under the Ministry of Social Security and Labour as a unit, providing electronic services. The whole data and all business integrations among internal and external systems are based on this application layer. Significant level of stability was reached by using technique of decomposition of this layer: all business processes and workflows are separated from web services. Integration layer consists of two independent application servers:
 - o BPEL application server
 - o WS application server

Decomposition allows increasing of any of these layers according to the system needs and/or requirements on a comparatively low cost. Virtualization of both of them decreases down time till 1 hour.

- **Web-tier layer:** contains static content that is provided for the end user. Along with static there also is web based user interface that allows to the end user to process its business action without knowing all the complexity of the system.

System architecture is created according benchmarking results and consultations with Oracle Corporation. According Oracle Corporation such system architecture allows to handle 300 requests per minute of unique user sessions (unique missed hits).

Real situation shows that this architecture holds about 100 unique missed hits per second (about 6000 requests per minute).

5.4. INTEROPERABILITY LAYER

Closer look at the integration layer shows the complexity of chosen architecture and number of used application servers.

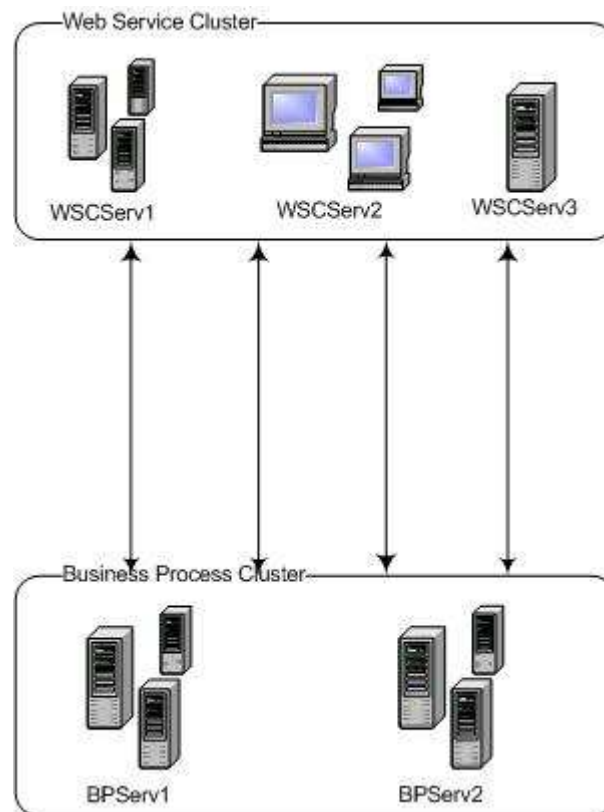


Figure 18: Interoperability layer decomposition.

Business process cluster holds all the workflow and process of business interaction among integrated systems. BPEL servers are using BPEL 1.0 specification and based on BPEL4WS standard. Orchestration and choreography among web services implemented using WSSE security standard.

WSSE security credentials are used when interacting systems are from different application layers. An example of such logics could be the situation when internal and external systems are integrated via this system.

Web service cluster holds bunch of web services logically grouped by systems. Logical grouping is essential for the systems maintenance and benchmarking. Initial solution was without such groupings, which lead to the complicity of result analysis and rapid system degradation (system failures). Each integrated system has its own container for web service deployments.

5.5. INTEROPERATING SYSTEMS

IT infrastructure of State Social Insurance Board under the Ministry of Social Security and Labour has variety of systems providing external and/or internal functionality. All these systems can be classified according their functionality and technology used:

- Internal systems:
 - o Axis over WSSE
 - o PL/SQL RMI, JAX-RPC.
 - o SAP.
- External Systems:
 - o BPEL, JAX-RPC.
 - o JAX-RPC over SSL.

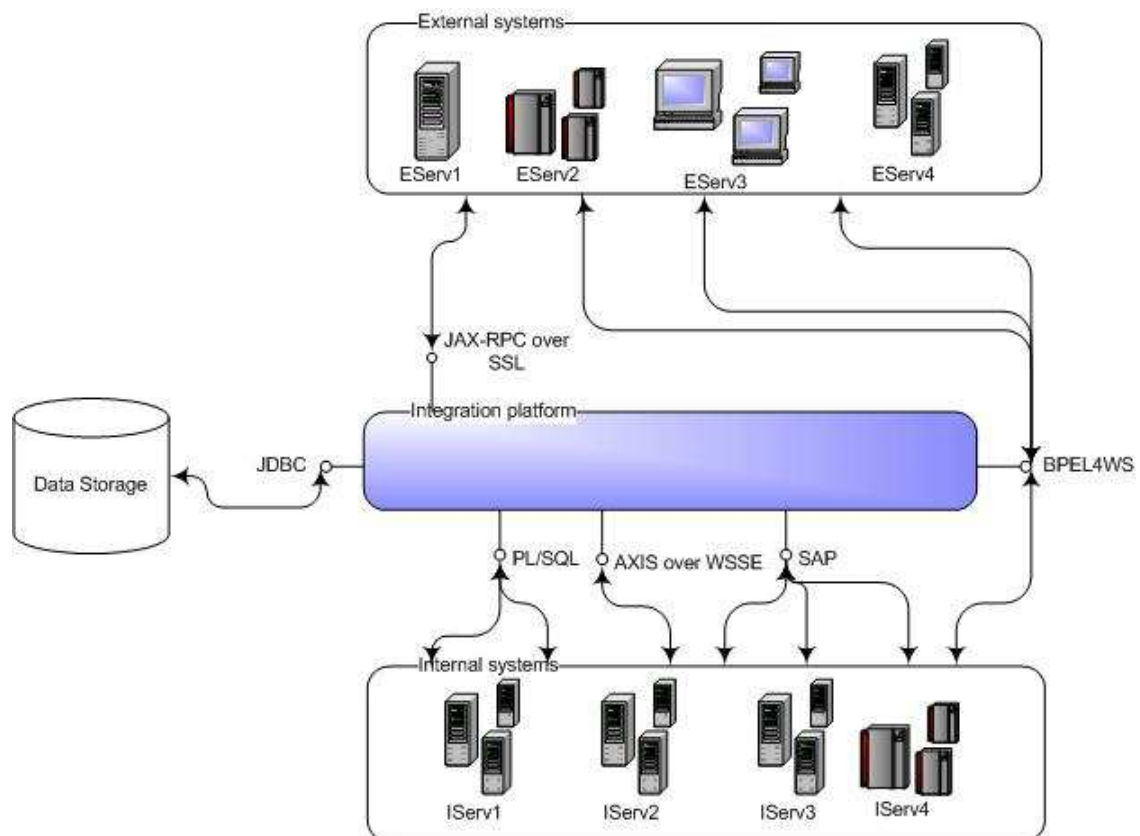


Figure 19: Interoperating systems and connection interfaces

Complexity and variety of technologies are successfully implemented using deep analysis, eServices, development strategies and results of successful implementations in variety of infrastructures analysis.

6. Conclusions and suggestions

Analysis of spread of technologies showed that there is strong need in cross-platform architecture that separates business logics and functionality into two separate layers. Business logics layer is one of the most important and one of the rapidly changing layer of interoperation and/or integration process (this architectural approach also known as web service component model (WSCM)) [50].

Among top ten integration technologies SOA provides best price/quality ratio. This architectural approach is maintained and supported by the most important players in current market – that brings opportunity of analysis of quality of services, provides support and other enterprise level features to increase productivity.

Oracle, IBM SAP and Microsoft provides best middleware solution to implement SOA. Choice of application strongly depends on background of Informational system and the number of interoperating systems.

Business practices show that to reach best performance/stability ratio the products of the same vendor must be used (to provide backward compatibility and support). Combination of different vendors complicates process of support and increases cost of maintenance. Strong position of open source solution products showed another point of view to the complex solutions – to increase development and migration possibility – decrease number of commercial products in your system. Problems with error fixes, slow reaction to the critical problems of users are moving them to an alternative – to open source solutions that would faster fixing simple problems or showing the problem causes (which can be easily modified by local IT specialists due to existence of open source).

Decomposition of SOA into several business layers brings simplicity and pure understanding of strong and weak sides of project architecture. More over such architectural model allows rapid adaptation for business needs: changing and/or substituting processing element with each other, modification of orchestration and choreography becomes less important for new solutions integration. Such approach stabilizes business processes and allows concentrating on separate details to increase performance of the whole system.

While developing and maintaining EDAS various problems of stability and performance were challenged. Majority of problems have been fixed and the whole productivity of the system have successfully been increased. Integration among different internal and external systems was done using WSCM architectural approach to minimize influence of business processes to entire stability of the system. Using SOA allowed implementing latest technologies of eServices development strategies.

7. References

1. The World Bank [viewed 2008.06.01]. Internet address:
<http://web.worldbank.org/WBSITE/EXTERNAL/TOPICS/EXTINFORMATIONANDCOMMUNICATIONANDTECHNOLOGIES/EXTGOVERNMENT/0,,menuPK:702592~pagePK:149018~piPK:149093~theSitePK:702586,00.html>
2. Internal Revenue Service. United States Department of the Treasury [viewed 2008.06.01]. Internet address: <http://www.irs.gov/taxpros/article/0,,id=109646,00.html>
3. “Online Availability of Public Services: How Is Europe Progressing?” Web Based Survey on Electronic Public Services Report of the 6th Measurement June 2006.
4. State Social Insurance Fund Board of the Republic of Lithuania under the Ministry of Social Security and Labor [viewed 2008.06.01]. Internet address:
<http://www.sodra.lt/index.php?cid=2303>
5. “The Future Development Strategy of Korea e-Government” [viewed 2008.07.14] Internet address: http://www.korea.go.kr/eng/Webzine/webzine.jsp?vol=200704&paper=korea_01&no=1197876844544
6. Dėl Elektroninių pranešimų kvalifikuoto elektroninio parašo taisyklių patvirtinimo [viewed 2008.06.01]. Internet address:
http://www3.lrs.lt/pls/inter3/dokpaieska.susije_l?p_id=312001
7. Dėl Elektroninių pranešimų kvalifikuoto elektroninio parašo taisyklių patvirtinimo. Susiję ES dokumentai [viewed 2008.06.01]. Internet address:
http://www3.lrs.lt/pls/inter3/dokpaieska.susije_l?p_id=312001&p_rys_id=-1
8. Elektroninio parašo įstatymo 4,8, 14, 16 straipsnių pakeitimo ir papildymo įstatymas. [viewed 2008.07.14]. Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=169558&p_query=&p_tr2=
9. Asmens tapatybės kortelės įstatymo 2, 4, 5 straipsnių pakeitimo bei papildymo ir įstatymo papildymo 11 straipsniu įstatymas [viewed 2008.07.14]. Internet address:
http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=323629&p_query=&p_tr2=
10. Buhalterinės apskaitos įstatymas [viewed 2008/07.14]. Internet address:
http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=308926&p_query=&p_tr2=
11. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. gruodžio 20 d. įsakymas Nr. V-665 „Dėl Elektroninės draudėjų aptarnavimo sistemos taisyklių patvirtinimo” [viewed 2008.07.18]. Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=312002&p_query=&p_tr2=

12. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. gruodžio 20 d. įsakymas Nr. V-666 „Dėl Elektroninių pranešimų kvalifikuoto elektroninio parašo taisyklių patvirtinimo“ [viewed 2008.07.18]. Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=312001&p_query=&p_tr2=
13. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 15 d. įsakymas Nr. V-256 „Dėl 1-SD pranešimo apie apdraustųjų valstybinio socialinio draudimo pradžią ir 2-SD pranešimo apie apdraustųjų valstybinio socialinio draudimo pabaigą pateikimo taisyklių patvirtinimo“ [viewed 2008.07.20]. Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=300228
14. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 26 d. įsakymas Nr. V-271 „Dėl 3C-SD pranešimo apie ūkinės bendrijos metinėje pelno mokesčio deklaracijoje nurodytos pajamų metinės sumos paskirstymą nariams pateikimo taisyklių patvirtinimo“ [viewed 2008.07.20]. Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=300959
15. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 26 d. įsakymas Nr. V-273 „Dėl 6-SD pranešimo apie draudėjo reorganizavimą pateikimo taisyklių patvirtinimo“ [viewed 2008.07.23]. Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=300960
16. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 15 d. įsakymas Nr. V-257 „Dėl 9-SD pranešimo apie motinai (įmotei), tėvui (įtėviui) arba vaiko globėjui suteiktas (atšauktas) atostogas vaikui prižiūrėti pateikimo taisyklių patvirtinimo“ [viewed 2008.07.23]. Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=300229
17. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 20 d. įsakymas Nr. V-261 „Dėl 12-SD pranešimo apie apdraustųjų nedraudiminių laikotarpius pateikimo taisyklių patvirtinimo“ [viewed 2008.07.23]. Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=300462
18. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 14 d. įsakymas Nr. V-251 „Dėl įgaliojimo pasirašyti pateikiamus socialinio draudimo pranešimus ir pranešimo apie įgaliojimo

- panaikinimą parengimo ir pateikimo tvarkos aprašo patvirtinimo“ [viewed 2008.07.23].
Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=300226
19. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 18 d. įsakymas Nr. V-258 "Dėl Valstybinio socialinio draudimo fondo valdybos direktoriaus 2004 m. birželio 21 d. įsakymo Nr. V-249 "Dėl apdraustųjų įskaitos duomenų koregavimo" pakeitimo" [viewed 2008.07.23].
Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=300230
 20. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. birželio 15 d. įsakymas Nr. V-254 "Dėl Valstybinio socialinio draudimo fondo valdybos direktoriaus 2004 m. gruodžio 28 d. įsakymo Nr. V-482 "Dėl Valstybinio socialinio draudimo pažymėjimo išdavimo ir keitimo taisyklių patvirtinimo" pakeitimo" [viewed 2008.07.23]. Internet address:
http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=300227
 21. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. liepos 4 d. įsakymas Nr. V-295 „Dėl SAM pranešimo apie apdraustuosius už ataskaitinį ketvirtį pateikimo taisyklių patvirtinimo“ [viewed 2008.07.23]. Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=301462
 22. Valstybinio socialinio draudimo fondo valdybos prie Socialinės apsaugos ir darbo ministerijos direktoriaus 2007 m. liepos 26 d. įsakymas Nr. V-331 „Dėl SAV pranešimo apie savarankiškai dirbančius asmenis pateikimo taisyklių patvirtinimo“ [viewed 2008.07.23]. Internet address: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=302454
 23. E-Government [viewed 2008.11.10]. Internet address: <http://en.wikipedia.org/wiki/EGovernment>
 24. Oracle9i JDBC Developer's Guide and Reference. Release 2 (9.2) [viewed 2009.01.14].
Internet address: http://download-west.oracle.com/docs/cd/B10501_01/java.920/a96654/ref.htm
 25. Information and Definition Schemas (SQL/Schemata). ISO/IEC 9075-11:2003, 2003, p. 1.
 26. Object-oriented analysis and design [viewed 2009.01.14]. Internet address:
http://en.wikipedia.org/wiki/Object-oriented_analysis_and_design
 27. Java and E-Services for Electronic Marketplaces. Giacomo Piccinelli. Hewlett-Packard Laboratories, Bristol (UK). 2000, p. 5.

28. XML in Data Management [viewed 2009.01.14]. Internet address:
<http://www.sciencedirect.com/science/book/9780120455997>
29. How to interpret COBOL statistics [viewed 2009.02.01]. Internet address:
http://weblog.infoworld.com/lewis/archives/2007/01/how_to_interpre.htmlsd
30. TIOBE index history of COBOL language [viewed 2009.02.01]. Internet address:
<http://www.tiobe.com/index.php/paperinfo/tpci/COBOL.html>
31. Web services [viewed 2009.02.07]. Internet address:
http://inews.webopedia.com/TERM/W/Web_services.html
32. Is there a market for web services? An analysis of web services directories. Christine Legner. Institute of Information Management, University of St. Gallen, Muller-Friedberg-Str. 8. Switzerland.
33. Report: Web services market to explode [viewed 2009.02.07]. Internet address:
<http://www.internetnews.com/dev-news/article.php/3413161>
34. XML and SOA. Tomas Erl. Internet address:
www.idealliance.org/proceedings/xml04/papers/91/xml_soa_paper.pdf
35. Service Oriented Architecture: Final Paper. Marc Strom, Kyle Estes, Kirti Kesavarapu, Brian Shipe [viewed 2009.03.13]. Internet address: <http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf>
36. Patterns: Service-Oriented Architecture and Web Services. First Edition (April 2004). International Business Machines Corporation 2004.
37. Attacking XML Security. Black Hat Briefings USA 2007. Brad Hill. iSEC Partners [viewed 2009.04.05]. Internet address:
http://www.isecpartners.com/files/iSEC_HILL_AttackingXMLSecurity_bh07.pdf
38. Web Services Security: What's Required To Secure A Service-Oriented Architecture? An Oracle White Paper. Marc Chanliau, October 2006 [viewed 2009.04.05]. Internet address: www.oracle.com/technology/tech/standards/pdf/security.pdf .
39. Web Services Security Policy Language (WS-SecurityPolicy). Version 1.1. July 2005.
40. Enterprise Service Bus. Sonic Progress [viewed 2009.04.10]. Internet address:
http://www.sonicsoftware.com/solutions/service_oriented_architecture/enterprise_service_bus/index.ssp
41. Microsoft ESD Guidance [viewed 2009.04.10]. Internet address: <http://blogs.msdn.com/fmarasco/default.aspx>.
42. Java Web Services. David Chappell, Tyler Jewell, O'Reilly. 2002
43. An Introduction to BPEL. Kumar Raj Moorthy [viewed 2009.04.12]. Internet address:
<http://www.developer.com/services/article.php/3609381>.

44. Trusted Web Service. Zhexuan Song. Fujitsu Laboratories of America, Inc. Maryland 2006.
45. Oracle SOA Suite [viewed 2009.04.12]. Internet address:
<http://www.networkcomputing.com/channels/appinfrastructure/showArticle.jhtml?articleID=181401184>
46. WebSphere Business Integration Server Foundation V5.1. Building and Deploying Service-Oriented Applications That Extend and Integrate Existing IT Assets. D.H. Brown Associates, Inc. IBM White Paper. 2004
47. SAP NetWeaver Introductions and Details [viewed 2009.04.20]. Internet address:
http://sapbrainsonline.com/Emerging_Tech/sap_netweaver.html
48. Microsoft BizTalk: Connecting Customers, Suppliers and Partner [viewed 2009.04.20]. Internet address: <http://www.web-enable.com/technology/BizTalkServer.asp>
49. Web Services for Remote Portlets [viewed 2009.04.24]. Internet address:
http://en.wikipedia.org/wiki/Web_Services_for_Remote_Portlets.
50. Web Service Component Model (WSCM) [viewed 2009.05.08]. Internet address:
http://www.service-architecture.com/web-services/articles/web_services_component_model_wscm.html