Aalborg Universitet

AALBORG UNIVERSITY
DENMARK

NI2S3- D1.1 NEC/SOA state of the art

Lund, David ; Heravi, Behzad ; Soltanpur, Cinna ; Pacyna, Piotr ; Rapacz, Norbert ; Pecorella, Tommaso; Rosi, Matteo; Sowa, Grzegorz; Stango, Antonietta

Publication date:
2010

Document Version
Early version, also known as pre-print

Link to publication from Aalborg University

Citation for published version (APA):
Lund, D., Heravi, B., Soltanpur, C., Pacyna, P., Rapacz, N., Pecorella, T., Rosi, M., Sowa, G., & Stango, A. (2010). *NI2S3- D1.1 NEC/SOA state of the art*. http://ni2s3-project.eu/publications/public-deliverables/ni2s3-deliverable1-1-nec-soa-state-of-the-art.pdf/view

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: April 25, 2024

European Commission
Enterprise and Industry

NI2S3

NET Information Integration Services for Security Systems

Project no.:	**225488**
Project full title:	**NET Information Integration Services for Security Systems**
Project Acronym:	**NI2S3**
Deliverable no.:	**D1.1**
Title of the deliverable:	**NEC/SOA state of the art**

Contractual Date of Delivery to the CEC:	M3 (30th September 2009)
Actual Date of Delivery to the CEC:	M12 (30th June 2010)

Editor:	**HWC**
Participant(s):	**AGH, UniFI, Comarch, CTIF**
Author(s):	**David Lund (HWC), Behzad Heravi (HWC), Cinna Soltanpur (HWC), Piotr Pacyna (AGH), Norbert Rapacz (AGH), Tommaso Pecorella (UniFI), Matteo Rosi (UniFI), Grzegorz Sowa (Comarch), Antonietta Stango (CTIF),**

Work package contributing to the deliverable:	WP1
Dissemination level:	PU
Nature:	R
Version:	1.0
Total number of pages:	**60**

Abstract:

This document presents the state of the Art in NEC and SOA and studies the envisaged use of SOA for the benefit of NEC. An initial analysis is made on key gaps with respect to security provisions for SOA in the NEC context. Previous and current projects are discussed together with applicable technologies and standards with their relevance to SOA and NEC and general NI2S3 concepts.

Table of Contents

Abstract: **1**
Executive Summary **9**
1 Introduction **10**
1.1 Background 10
1.2 NI2S3 Objectives 10
2 NEC & SOA State of Art **12**
2.1 Network Enabled Capability 12
2.2 Service Oriented Architecture (SOA) 15
2.3 Other technologies (.NET WCF) 17
3 NEC and SOA integration **19**
3.1 Achieving NEC by SOA 19
3.2 Security Knots 21
4 Projects and initiatives **22**
4.1 R&D Projects 22
5 Research areas **30**
5.1 Security of SOA and Web Services 30
5.1.1 Basic building blocks of Web Services Security 30
5.1.2 Web Services security standards 31
5.1.3 Other WS standards relevant for NI2S3 34
5.2 SOA reliability and dependability - open issues and challenges 35
5.2.1 Dependability ontology 35
5.2.2 Attributes of dependability 36
5.2.3 Threats to dependability 37
5.2.4 Means to preserve dependability 38
5.2.5 Accountability extensions to dependability 39
5.3 Challenges and open issues for SOA dependability in NEC context 40
5.3.1 Fault tolerance in SOA 40
5.3.2 Replication 41
5.3.3 Middleware extensions 41
5.3.4 Real time SOA 41

5.3.5 Evolving architecture 42
5.3.6 Evaluation of architectures 42
5.3.7 Instrumentation (Tracking and Monitoring) 42
5.4 VALIDATION AND COMPLIANCE TESTING METHODS 42
5.4.1 Testing 42
5.4.2 Software security testing 44
5.4.3 Technology for black box testing 45
5.4.4 Known tools and software 50
6 Summary and Conclusion 55
7 References 56

Abbreviations

With a breadth of activities ongoing in Europe and beyond, we intend to form a common taxonomy for use throughout the project.

Acronym	Meaning
AAA	Authentication, Authorization, and Accounting
ADABTS	Automatic Detection of Abnormal Behaviour and Threats in crowded Spaces
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASP	Active Server Pages
BNF	Backus-Naur Form
BPEL	Business process execution language
BPM	Business Process Management
C4ISTAR	Command, Control, Communications, Computers, Intelligence, Surveillance, Target Acquisition and Reconnaissance
CCTV	Closed Circuit Television
CIIP	Critical Information Infrastructure Protection
CLR	Common Language Runtime
CMS	Content Management System
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CTMF	Conformance Testing Methodology and Framework
DETECTER	Detection Technologies, Counter-Terrorism Ethics, and Human Rights
DoDAF	Department of Defence Architecture Framework
ebXML	Electronic Business using eXtensible Markup Language
EC	European Commission
EFS	Evolutionary Fuzzing System
eID	electronic identity

eIdM	Electronic Identity Management
ENISA	European Network and Information Security Agency
ESB	Enterprise Service Bus
ETSI	European Telecommunications Standards Institute
EU	European Union
EU-SEC II	Coordinating National Research Programmes and Policies on Major Events Security
FSM	Finite State Machine
FT	Fault Tolerance
FTP	File Transfer Protocol
GPF	General Purpose Fuzzer
GPL	GNU General Public License
GSM	Global System for Mobile communications
HTTP	HyperText Transfer Protocol
IAM	Identity and Access Management
IEC	International Electrotechnical Commission
IEV	International Electrotechincal Vocabulary
IFIP	International Federation for Information Processing
Indect	Intelligent Information System Supporting Observation, Searching and Detection for Security of Citizens in Urban Environment
ITU-T	International Telecommunications Union-Telecommunication
J2EE	Java Platform, Enterprise Edition
LDAP	Lightweight Directory Access Protocol
LLAMA	The inteLLigent Accountability Middleware Architecture
MART	Mean Active Repair Time
MASTER	Managing Assurance, Security and Trust for sERvices
MoD	Ministry of Defence
MoDAF	Ministry of Defence Architecture Framework

MSMQ	Microsoft Message Queuing
MTBF	Mean Time Between Failure(s)
MTFF	Mean Time to First Failure
MTTF	Mean Time To Failure
NAF	NATO Architecture Framework
NEC	Network Enabled Capability
NECTISE	Network Enabled Capability through Innovative Systems Engineering
NIS	Network and Information Security
NIST	National Institute of Standards and Technology
NSOV	NATO Service Oriented View
OASIS	Organization for the Advancement of Structured Information Standards
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PICOS	Privacy and Identity Management for Community Services
PRIME	Privacy and Identity Management for Europe
PrimeLife	Bringing sustainable privacy and identity management to future networks and services
PROTOS	Project Security Testing of Protocol Implementations
QoS	Quality-of-Service
RST	Request Security Token
RT	Real Time
SABSA	SHERWOOD APPLIED BUSINESS SECURITY ARCHITECTURE
SAML	Security Assertions Markup Language
SAMURAI	Suspicious and Using a netwoRk of cAmeras for sItuation awareness Enhancement
SERICOM	Seamless Communication for Crisis Management
SGML	Standard Generalized Markup Language
SIP	Session Initiation Protocol

SLA	Service level agreements
SLO	Service Level Objectives
SME	Small and Medium Enterprises
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SRS	Shared Registry System
STORK	Secure idenTity acrOss boRders linKed
STS	Security Token Services
SUBITO	Surveillance of unattended baggage and the identification and tracking of the owner
SWIFT	Secure Widespread Identities for Federated Telecommunications
TEDS	TETRA Enhanced data service
TETRA	TErrestrial Trunked Radio
TOGAF	The Open Group Architecture Framework
UDDI	Universal Description Discovery and Integration
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WCF	Windows Communication Foundation
WS	Web Services
WS-*	Web services specifications
WSDL	Web Services Description Language
WSE	Web Services Enhancements
WSLA	Web Service Level Agreements

XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

Executive Summary

This document presents the state of the Art in NEC and SOA and studies the envisaged use of SOA for the benefit of NEC. An initial analysis is made on key gaps with respect to security provisions for SOA in the NEC context. Previous and current projects are discussed with their relevance to SOA and NEC and general NI2S3 concepts.

It is clear through the reference material in this document that the state of the art in SOA itself is fairly well advanced with several examples of commercial use of SOA based infrastructure deployments. The research field is very active with regard to SOA with activity generally focused toward specific aspects such as security, dependability and compliance. Many relevant activities have been identified in this document, which can form major input basis to the work of NI2S3.

With regard to NEC there is material mainly published through larger national defense frameworks which are primarily encapsulated within Enterprise Architecture Frameworks such as MoDAF[9], DoDAF[7] and NAF.

No detailed material is found which uses NEC for the use by CIIP [3], although SOA is an enabling architecture. The Enterprise Architecture Frameworks do, however, present a broad application onto architectures which are not necessarily military based, although no material has been identified which deals explicitly with the concept of using SOA for NEC in the context of CIIP. This issue is expanded in the companion deliverable D1.2.

1 Introduction

NI2S3 is a project funded by European Commission FP7 programme. This document presents deliverable D1.2 as a part of Workpackage 1 with the primary aim to assess the State of the Art in Network enabled Capability (NEC) and Service Oriented Architectures (SOA)

1.1 Background

Critical infrastructures are often protected by several protection systems of various types. In such complex systems, the situation awareness is the key to the success in securing the infrastructure. Unfortunately, protection systems often act independently, and therefore can fail at discovering and reacting to minor alarms. NEC Information and Integration Services (NEC) permit to create situational awareness and allow to share the view of the protected infrastructure, thus facilitating the decision making. The NEC methodology, originally applied in some defence applications, can also be used for the protection of infrastructures in civil applications, however a methodology for developing effective protection systems needs to be elaborated. The aim of the NI2S3 Project is to address this vacancy and to come up with a reference methodology for building critical infrastructure protection systems based on of NEC framework.

1.2 NI2S3 Objectives

Critical infrastructures are central to the sustainable development in societies and economies. The existing critical infrastructures have been evolving for a long time and have become large. Protection systems for such infrastructures have usually been designed for some specific purpose, and are usually operated as independent systems. Along time, some new, increasingly sophisticated capabilities have been added to the protection systems.

Complex interactions between the elements of a critical infrastructure indicate that there is also a need to deploy a corresponding infrastructure protection system, which is capable of extending security control to all elements of the protected system, and which is, at the same time, capable of maintaining a global view of the infrastructure. Unfortunately, one of the concerns with the networked protection systems is related to the complexity of interactions and to the amount of exchanges, during the acquisition, transmission, aggregation and processing of data pertaining to the state of the elements in the protected infrastructure. The amount of data, its different type and origin, can quickly become overwhelming to an aggregation and processing system, thus making any systematic correlation and inference about the state of the infrastructure quite infeasible. As a result, protection systems are becoming incapable of ensuring appropriate security levels. Such situation requires an approach, which is different than what is commonly supported today.

The key objective of the NI2S3 project is to research and implement a reference methodology for developing security systems based on NEC Information and Integration Services. The security systems must be capable of collecting and processing information from many heterogeneous sources in order to build up or improve situation awareness of critical infrastructures.

More specifically, the NI2S3 Project aims:

- to provide a definition and a design of an NI2S3 critical infrastructure protection system regarding the security, resiliency and availability of the subject infrastructure,
- to define performance indicators and tools for system validation,
- to develop a technology for the evaluation of the performance, robustness and reliability of such a protection system,
- to develop a NI2S3 application demo.

The resulting protection system should involve all the necessary components and tools to acquire, exchange and process the state monitoring information. It should rely on the continuous feeding of the information, in order to ensure that it arrives at the right place, right time, preferably in the form, which makes it quickly usable for the intended purpose, and which can result in appropriate and timely actions. NI2S3 Project will ensure that the prospective protection system is error-proof, in what concerns vulnerabilities. As an example, the protection system must not react in ways that may lead to erroneous, inadequate or disproportional system reactions. Instead, the NI2S3 system has to provide information at different granularity levels in a timely manner to plan, direct and control all operational activities pertaining to critical infrastructure protection.

This document initially presents the state of the Art in NEC and SOA and studies the envisaged use of SOA for the benefit of NEC. An initial analysis is made on key gaps with respect to security provisions for SOA in the NEC context. Previous and current projects are discussed with their relevance to SOA and NEC and general NI2S3 concepts. A considerable section presents some fo the wider and active research areas, primarily focuses around web services with a more general application within SOA. A comprehensive list of applicable references is given together with list of standards from which all material presented in this document is based upon.

2 NEC & SOA State of Art

Within this section, we discuss NEC and SOA in their own separate and individual contexts.

Section 2.1 presents the highest level conceptual definition of NEC. Please note that D1.2 explains various publicly available Enterprise Architecture Frameworks from which specific implementations can yield. A further inclusion of Enterprise Security Architectures is also presented in D1.2.

Section 2.2 describes the technical state of art and capabilities of Service Oriented Architecture and those components which are applicable for implementation of an NEC focussed system.

2.1 Network Enabled Capability

Network Enabled Capability is a term defined originally by the UK Ministry of Defense (MoD) [MODNEC] [15] which promotes the extended use of communication and data network technologies for providing an extension to the capability of critical operations.

The earliest reference to NEC is given as follows

"The ability to gather knowledge; to share it in a common and comprehensible form with our partners; to assess and refine it to turn into knowledge; to pass it to the people who need it in an edited, focussed form; and to do it in a timescale necessary to enable relevant decisions to be made in the most economic and efficient manner" - [DCDS(EC) 8 Nov 01]

More recent and simplistic

"Network Enabled Capability (NEC) is about the coherent integration of sensors, decision makers and weapon systems along with support capabilities" – [22]

The term NEC is originally defined and largely associated with its use in the military domain. For NI2S3, we intend to reapply NEC techniques for use in the management of critical infrastructures. Figure 1 illustrates how technology can be applied for improved capability.

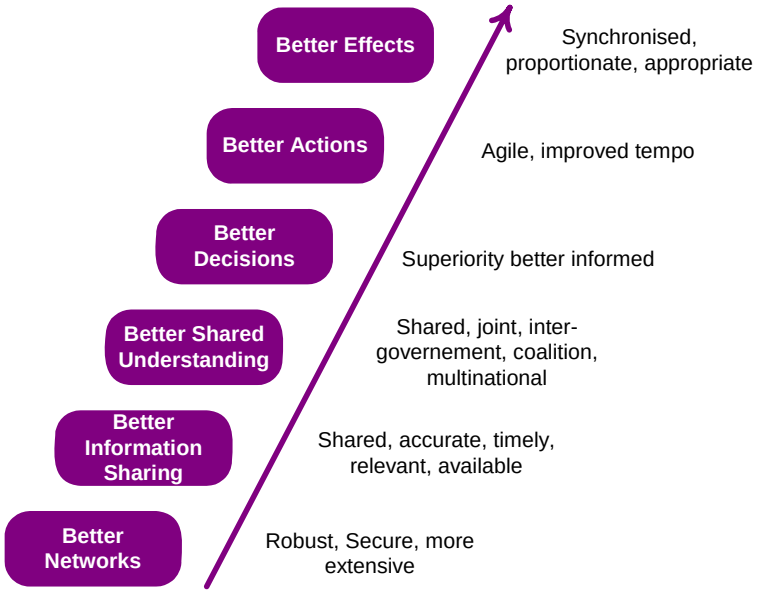

Figure 1 Improvements in capability brought by NEC [15] [MODNEC]

NEC is a term that intentionally fuses together people, networks and information. The concept draws upon managerial and social aspects which are needed to build a capability and makes use of modern and advancing technology to achieve that fusion. Figure 2 illustrates the conceptual grouping of people, information and networks for a combined, joint and consolidated resultant capability.

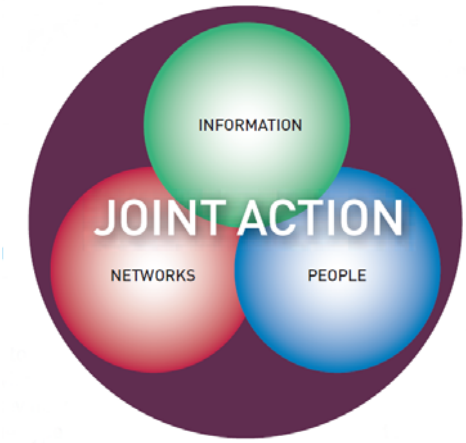

Figure 2 Key components of NEC [15][MODNEC]

NEC describes a broad class of approaches which were originally defined for military and homeland security in the context of operations that are enabled by the networking of those components which develop a core and consolidated force. NEC concepts can be understood by focusing on the following 3 relationships, that take place simultaneously in and among the physical-, the information-, and the cognitive domains.

- **Physical Domain.** The physical domain is where physical platforms and the communications networks that connect them reside. Comparatively, the elements of this domain are the easiest to measure. Performance of a critical operation has traditionally been measured primarily in this domain due to the involvement of mainly physical humans and increasingly automated physical equipment. In NEC, where communications move towards a more automated and connected physical approach, all elements within the physical domain are robustly networked achieving secure and seamless operation.
- **Information Domain.** The information domain is the domain where information is stored, manipulated, shared and viewed. It is the domain that facilitates the communication of information among key sources, consumers, processors and operators of the system. Consequently, and increasingly, the information domain must be protected and defended to enable a system or service to retain its capability to perform and react. The service has the capability to collect, share, access, collaborate, analyze, and protect information, achieving an information advantage over changing and adverse operational conditions.
- **Cognitive Domain.** The cognitive domain is where high-quality situational awareness is associated for the use by the management / commander staff to take decisions and implement those through synchronized operations.

2.2 Service Oriented Architecture (SOA)

The Service Oriented Architecture (SOA) is a network-enabled solution that has the potential to combine assets (software resources, people, equipment and processes) to provide capability; that is, the ability to achieve a mission objective.

SOA is an information technology approach or strategy in which applications make use of (perhaps more accurately, rely on) services available in a network. The use of services provides a distributed computing approach for integrating extremely heterogeneous applications over the network.

The functions of an application or system (including legacy systems) can be easier to access as a service in an SOA than in some other architecture. So integrating applications and systems can be much simpler.

The Web service specifications are completely independent of programming language, operating system, and hardware.

The technology is based on open technologies such as:

- eXtensible Markup Language (XML)
- Simple Object Access Protocol (SOAP)
- Universal Description, Discovery and Integration (UDDI)
- Web Services Description Language (WSDL)

Using open standards provides broad interoperability among different vendor solutions.

XML

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

SOAP

Simple Object Access Protocol (SOAP) is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts:

1. an envelope that defines a framework for describing what is in a message and how to process it,
2. a set of encoding rules for expressing instances of application-defined data types, and
3. a convention for representing remote procedure calls and responses.

UDDI

Universal Description, Discovery, and Integration (UDDI) protocol is an approved OASIS Standard and a key member of the Web services stack. It defines a standard method for publishing and discovering the network-based software components of a service-oriented architecture (SOA).

WSDL

Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services).

Figure 3 shows an example of SOA Architecture. The stack includes all of platforms and tooling available for an enterprise SOA, but a simple architecture will not necessarily include all elements of the stack. The Enterprise Service Bus (ESB) is the only necessary component to have a SOA.

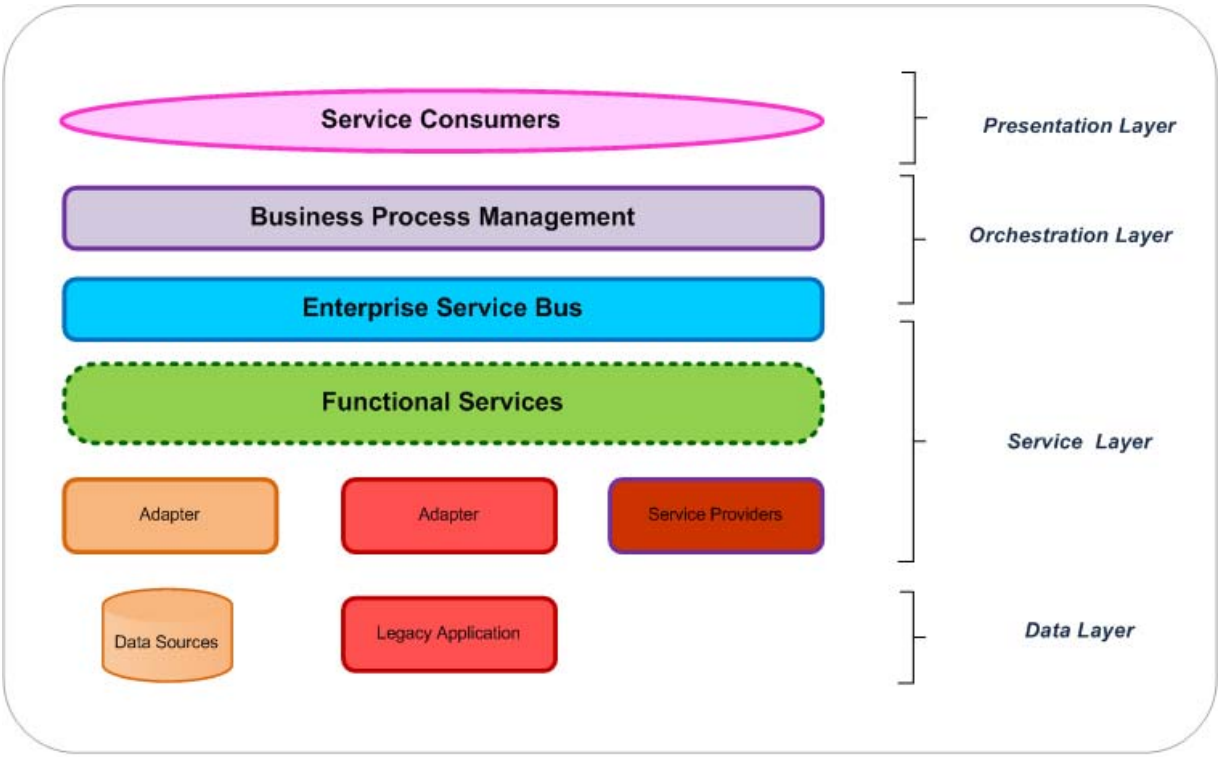

Figure 3 SOA Architecture

Functional Services

These are the atomic services coming from the adapters or from the service providers. The adapter is a layer or an interface between an old system (for example legacy systems) and the ESB.

ESB

An Enterprise Service Bus is a software architecture which provides fundamental services for SOA architectures. The most important functions supported are:

- Invocation
- Message Routing
- Mediation
- Messaging
- Service Orchestration
- Security
- Management

BPM

The Business Process Management (BPM), is a software component that manages and orchestrates human tasks and system services in a systematic way to streamline business processes. Business process execution language (BPEL) is the de-facto standard for automating processes in a SOA.

Service Consumers

The service consumers or end user can be a different type of systems. For example a Portal, Business Intelligence application, or a system for decision support.

2.3 Other technologies (.NET WCF)

Windows Communications foundation (WCF) is a Microsoft technology for SOA. First released as part of the .NET Framework 3.0 in 2006, WCF simplifies development of connected applications through a new service-oriented programming model. WCF supports many styles of distributed application development by providing a layered architecture. At its base, the WCF channel architecture provides asynchronous, untyped message-passing primitives. Built on top of this base are protocol facilities for secure, reliable, transacted data exchange and broad choice of transport and encoding options.

The typed programming model (called the *service model*) is designed to ease the development of distributed applications and to provide developers with expertise in ASP.NET Web services, .NET Framework remoting, and Enterprise Services, and who are coming to WCF with a familiar development experience. The service model features a straightforward mapping of Web services concepts to those of the .NET Framework common language runtime (CLR), including

flexible and extensible mapping of messages to service implementations in languages such as Visual C# or Visual Basic. It includes serialization facilities that enable loose coupling and versioning, and it provides integration and interoperability with existing .NET Framework distributed systems technologies such as Message Queuing (MSMQ), COM+, ASP.NET Web services, Web Services Enhancements (WSE), and a number of other functions.

WCF addresses a range of challenges for communicating applications. Three things stand out, however, as the most important aspects of WCF:

- Unification of existing .NET Framework communication technologies.
- Support for cross-vendor interoperability, including reliability, security, and transactions.
- Explicit service orientation.

Because WCF's fundamental communication mechanism is SOAP-based Web services, WCF-based applications can communicate with other software running in a variety of contexts. An application built on WCF can interact with all of the following:

- WCF-based applications running in a different process on the same Windows machine.
- WCF-based applications running on another Windows machine.
- Applications built on other technologies, such as J2EE application servers, that support standard Web services. These applications can be running on Windows machines or on machines running other operating systems.

While WCF introduces a new development environment for distributed applications, it is designed to interoperate well with the non-WCF applications. There are two important aspects to WCF interoperability: interoperability with other platforms, and interoperability with the Microsoft technologies that preceded WCF[38] [39].

3 NEC and SOA integration

There is very little available material which studies in detail, the use of SOA for the NEC approach. The most useful material is that produced by the NECTISE project which makes an initial basis through publication, most of which is only 1-2 years old. The roadmap for the NECTISE project would expect that some of this activity is ongoing in the academic domain although at the time of writing there is no formal liaison through which we can gain access to their work.

Revision 3 of the NATO architecture framework [23] explicitly makes provision for the use of SOA. Revision 3 supposedly builds upon MoDAF[9] and DoDAF[7] together with industrial experiences and explicitly defines the NATO Service Oriented View (NSOV) for achievement of this purpose. D1.2 explains MoDAF and DoDAF.

The nature of SOA matches primarily to the flexibilities needed for the orchestration of NEC capabilities.

SOA primarily decentralises middleware. Whilst this can provide a significant overhead on each service, the benefits are justified as described in [40]:-

- Loose Coupling is an architectural property exhibited by services that makes them independent from other components in the system.
- Defining services by interface, including data exchange and behaviour (pre/post conditions). This allows implementations to be interchangeable, offering dependability, availability and scalability by replication of services.
- Interface definitions also support late binding of services and resources, which supports evolvable systems by changing implementations to improve performance such as speed and accuracy.
- Reusable services. Loose coupling promotes the reuse of services in new contexts not previously envisaged.

Inter-organisational. By using a loosely coupled system, an application or process would be able to use services developed outside organizational bounds. The integration of services is achieved through the definition of procedures and workflows. This can support ultra-late binding, by selecting services implementations at the point of service execution.

3.1 Achieving NEC by SOA

SOA is defined as a framework for the distribution and use of networked services. This generally makes no specific assumption on the platforms upon which type of device or where the services are deployed, nor does SOA consider the types of data aggregation which may be required to be presented to enable a specific capability

For NEC, the following key requirements are placed on the SOA ‘toolbox’ :-

- Availability - the probability that a service is present and ready for use;
- Reliability - the capability of maintaining the service and service quality;

- Safety - the absence of catastrophic consequences;
- Confidentiality - information is accessible only to those authorized to use it;
- Integrity - the absence of improper system alterations;
- Maintainability - to undergo modifications and repairs;
- Cost - the price customers are willing to incur to obtain a level of service.

In addition to these, the consideration should be made of technologies which already play a part in critical infrastructures and those emerging to become powerful technologies for enhancing capability. Areas of concern include wireless technologies whereby operatives and/or devices need to be mobile in the critical environment [40].

Such technologies include:-

- Sensor Networks – localized cooperating devices which can be used for gathering sensory information in a distributed form. SOA may provide too much heavy functionality for small devices such as these
- Professional Mobile Radio (i.e. TETRA) – TETRA is widely used in critical environments. TETRA has primarily provided a voice service to date although there is currently a fast uptake of data services. The TETRA Enhanced data service (TEDS) is now available which can provide wideband data rates up to 600kbps. Providing SOA over TETRA can be limited by available bandwidth and latency.
- Existing Cellular (GSM, 3G and beyond) – Similar to TETRA, with a higher data capability but lesser in terms of reliability

3.2 Security Knots

The NEC concept does not define specific security requirements or constraints within the overall framework. However, the different 'capabilities' defined within a particular NEC driven framework may have different requirements. The capabilities specific to NI2S3 shall be define during WP2.

SOA already provides various standardised hooks for provision of a varied security topology although many of these hooks are abstract and require refinement during a specific implementation and subsequent orchestration. Key items of flexibility across the NEC, potentially provided by SOA are as follows in terms of both the individual service and the connectivity required to access that service:

- Reliability

  Section 5.1 describes specific SOA building blocks which are available for extension for a particular capability.

- Dependability

  Section 5.2 describes open issues with regard to dependability of web services. This includes somewhat the Availability criteria.

- Trust

  Section 5.1 also describes specific SOA building blocks for trusted message exchange. However, this can be restrictive in terms of trust of the service itself.

- Compliance

  A particular NEC capability may fall within well defined and strict operating criteria. Should the real operating performance fall short of such criteria, the resulting capability may be invalidated in terms of inaccurate provision of outputs which may have significant effects on decision making or be later problematic should an issue arise later.

  Operation of the capability and associated compliance checking may have to stand up to a court of law.

  All criteria related to Availability, Reliability, Safety, Confidentiality, Integrity, Maintainability and Cost must generally have forms of operation monitoring and compliance checking.

  Section 5.4 describes methods validation and compliance

The following sections describe the currently available technologies for these in further detail

4 Projects and initiatives

4.1 R&D Projects

In this section we summarise R&D projects and activity that may be relevant to this research. Topics covered include monitoring and surveillance, assurance and trust, privacy, privacy and electronic identity management, and Communication for crisis management. Whilst these project may not immediately approach SOA and NEC, it is considered that at least their general concepts are applicable for consideration in NI2S3.

Indect

Intelligent Information System Supporting Observation, Searching and Detection for Security of Citizens in Urban Environment

The Indect Integration Project has the objective to develop a platform for:

- automatic detection of threats and recognition of abnormal behavior or violence through processing video surveillance information.
- registration and exchange of operational data and acquisition of multimedia content,

To achieve this goal Indect aims:

- to develop the prototype of an integrated, network-centric system supporting the operational activities of police officers, providing techniques and tools for observation of various mobile objects,
- to develop a new type of search engine combining direct search of images and video based on watermarked contents, and the storage of metadata in the form of digital watermarks.

The Indect project declares the following results:

- a trial installation for the monitoring and surveillance system in various points of city agglomeration,
- implementation of a distributed computer system that is capable of acquisition, storage, on-demand data sharing as well as intelligent processing,
- construction of a search engine for fast detection of persons and documents based on watermarking technology and utilizing comprehensive research on watermarking technology used for semantic search,
- elaboration of Internet based intelligence gathering system.

More information can be found at: http://www.indect-project.eu/

MASTER

Managing Assurance, Security and Trust for sERvices

The MASTER project focuses on different levels of trust between entities and provides methodologies and infrastructures that facilitate the monitoring, enforcement, and audit of quantifiable indicators on the security of a process.

The MASTER project aims to provide models, technology, and tools to define policies, goals and performance indicators from a security, trust, and assurance perspective, and to map goals and indicators across levels of abstractions, as well as enforce such policies, and allow for visibility and audit ability of goals, indicators, and compliance with policies.

MASTER objectives address the following levels of complexity:

- Inside Single Trust Domain to support the management of assurance of the single provider. This includes protection against insider fraud as well as assessment and integration of existing security mechanisms.

Distributed Multiple Domains solutions needed to establish security between different trust domains building upon the results achieved for the single trust domain. This tackles the challenge of providing end-to-end security within a loose federation of mutually distrusting organizations.

Some potential output relevant to the objectives of NI2S3 may include: assurance of the security levels, trust levels and regulatory compliance of dynamic service-oriented architecture.

More information on MASTER project can be found at: http://www.master-fp7.eu .

PRIME

Privacy and Identity Management for Europe

The project aimed at addressing the means for the exchange of personal data, referred to as the so-called partial identities, which may convey sensitive personal data, such as patient health data, employment data, banking card data for the use in application areas such as public services as well as in public security (e.g. in border controls). PRIME focused on solutions for privacy, supported end-users' sovereignty over their private sphere and enterprises' privacy compliant data processing.

PRIME developed a working prototype of a Management System. To foster market adoption, partial solutions for managing identities were demonstrated in real Communication, Airline and Airport Passenger Processes, Location and Collaborative e-Learning.

This project has terminated in 2008, but it is now continued as PrimeLife project.

More information on PRIME can be found at: https://www.prime-project.eu/

PrimeLife

Bringing sustainable privacy and identity management to future networks and services

The PrimeLife project addresses the need to protect autonomy of human individuals and to retain self-control over their personal information, irrespective of their activities.

- The first challenge is about how to protect privacy in emerging Internet applications such as collaborative scenarios and virtual communities.

- The second challenge is how to maintain life-long privacy.

PrimeLife aims to resolve the core privacy and trust issues pertaining to these challenges.

More information can be found at http://www.primelife.eu/

SWIFT

Secure Widespread Identities for Federated Telecommunications

The SWIFT project leverages technology for identity management as a key to integrate service and transport infrastructures for the benefit of users and the providers. It focuses on extending identity functions and federation to the network while addressing usability and privacy concerns.

The scope of the project covers transport and services strata across all protocol layers, with the user's identity being intrinsic to the control, data and management plane protocols. Specifically, technological advances and breakthroughs are targeted for:

- Vertical integration of identity, privacy, trust and security across layers with the use of protocols, addressing schemes and inter-layer interfaces that provide controlled privacy for the user.
- identity-centric user schemes supporting different levels of information access control, both policy as well as credential-based with well-defined privacy rules about who can change or know the data handled.
- Methods and techniques on how users are identified and located, but at the same time remain pseudonymous at all layers based on preferences set by the users and their context,
- Techniques for name and identifier resolution across very heterogeneous namespaces.
- Identity-based mobility solution through adaptation of mobility protocols to the user's "moving identities" across devices, services and networks,
- An Identity Management Platform providing a common framework and APIs for accessing identity attributes across services and networks in a controlled way enabling user privacy mechanisms including specific APIs, such as for an Identity Broker.
- Mapping new identity techniques to existing technology (SIM cards, etc), and eIdM and AAA solutions to accommodate Identity Management. Specification and validation of extensions or modifications of existing solutions to support SWIFT vision.

More information can be found at: http://www.ist-swift.org

STORK

Secure idenTity acrOss boRders linKed

The STORK Project aims at developing a series of pilot projects which should be available to citizens of several European countries, using the identification/authentication means preferred by the governments of those countries. The STORK project makes it easier for citizens and

businesses to access online public services across borders by developing and testing common specifications for mutual recognition of national electronic identity (eID) between the participating countries. It will approach these objectives by:

- Developing common rules and specifications to assist mutual recognition of eIDs across national borders;
- Testing, in real life environments, secure and easy-to-use eID solutions for citizens and businesses;
- Interacting with other EU initiatives to maximize the usefulness of eID services.

STORK will focus on pragmatic eID interoperability solutions, implementing several pilot cross-border eID services chosen for their high impact on everyday life.

Currently the following pilots are under way:

- Pilot1: Cross border authentication platform - for electronic services,
- Pilot2: Safer Chat - To promote safe use of the Internet by children and young people,
- Pilot3: Student Mobility - To help people who want to study in different Member States,
- Pilot4: Electronic Delivery - To develop cross-border mechanisms for secure online delivery of document,
- Pilot5: Change of Address - To assist people moving across EU borders.

It is expected that the pilot programme will:

- contribute to accelerating the deployment of eID for public services, while ensuring co-ordination between national and EC initiatives in the field, and support federated eID management schemes across Europe based on open standard definitions where appropriate; and
- test, in real life environments, secure and easy-to-use eID solutions for citizens and businesses, in particular SMEs and government employees at relevant levels (local, regional, national and cross-border levels).

More information can be found at: http://www.eid-stork.eu/

PICOS

Privacy and Identity Management for Community Services

The PICOS project is developing and building a state privacy and identity management aspects of community services and applications on the Internet and in mobile communication networks. The PICOS approach to trustworthy on-line community collaboration addresses the following issues:

- Trust, Privacy and Identity issues in new context communication services, especially community-based services,
- Support for acceptable, trustworthy, open, scalable methods.

The work is focusing on platform design and prototype development in order to create interoperable, open, privacy respecting identity and trust management tools.

More information can be found at: http://picos-project.eu

ADABTS

Automatic Detection of Abnormal Behaviour and Threats in crowded Spaces

ADABTS project aims to facilitate the protection of EU citizens, property and infrastructure against threats of terrorism, crime and riots by the automatic detection of abnormal human behavior. In order to achieve it, ADABTS aims to develop models for abnormal and threat behaviors and algorithms for automatic detection of such behaviors, as well as for deviations from normal behavior in surveillance data.

Also, ADABTS aims to develop hardware, in order to enable such systems. The proposed system tracks and classifies objects in the scene and analyses their behavior according to specified alarm criteria.

More information can be found at:

http://cordis.europa.eu/fetch?CALLER=FP7_SECURITY_PROJ_EN&ACTION=D&DOC=37&CAT=PROJ&QUERY=0123e36de3ce:4312:22d3b7d9&RCN=91158

SAMURAI

Suspicious and Using a netwoRk of cAmeras for sItuation awareness Enhancement

The SAMURAI project aims to develop robust moving object, segmentation, categorisation and tagging in video captured by multiple cameras from medium-long range distance, e.g. identifying, monitoring and tracking people with luggage between different locations at an airport. Automated focus of attention and identification in a distributed sensor network that includes fixed and mobile cameras, positioning sensors, and wearable audio/video sensors.

Global situational awareness assessment and image retrieval of objects by types, movement patterns with incidents across a distributed network of cameras is in scope of the project.

Online adaptive abnormal behavior monitoring for profiling and inference of abnormal behaviors or events captured by multiple cameras.

The project aims to incorporate methods for feeding back into the algorithm human operator's evaluation on any abnormality detection output in order to guide and speed up the incremental and adaptive behavior profiling algorithm.

SAMURAI is developing technology that can be interfaced with the existing CCTV systems. It aims to allow for prevention and rapid-response to events as they unfold.

More information can be found at: http://www.samurai-eu.org/

SECRICOM

Seamless Communication for Crisis Management

The SECRICOM project aims to solve problems of crisis communication infrastructures through the creation of pervasive and trusted communication infrastructure and bringing interconnectivity between different networks of the following characteristics:

- provisioning of true collaboration and interworking of emergency responders,
- seamless support for different user traffic over different communication bearers,
- instant information gathering and processing focusing on emergency responders.

The Project aims to add new functions using distributed IT systems based on an SDR secure agents infrastructure.

More information can be found at: http://www.secricom.eu/

SUBITO

Surveillance of unattended baggage and the identification and tracking of the owner

The SUBITO project aims to research and develop automated detection of abandoned luggage, fast identification of the individual responsible and the tracking of their subsequent path.

The consortium plans to develop integrated threat detection system for a robust, timely alert to security personnel. A system will be capable of distinguishing between genuine threats and false alarms in order to alert the user to high priority situations.

The detection of unattended goods and of its owner will be focused on the automated real time detection of abandoned luggage or goods and the fast identification of the individual who left it

The key design drivers include assessment of the situations faced in such scenarios, and the existing security equipment available that will support the automatic operation of such functionality. To achieve the above, the SUBITO project brings in:

- expertise in state-of-the-art processing and detection and tracking algorithms,
- sensor data processing, sensor design and sensor systems integration.

SUBITO addresses objectives which are similar to those of NI2S3 in the area of detection of threats by means of surveillance.

More information can be found at: http://www.subito-project.eu/

DETECTER

Detection Technologies, C and Human Rights

Police and intelligence have recently increasingly focused on methods of preventing future attacks, and not just on identifying the perpetrators of offences already committed. Preventive police work includes the use of detection technologies. These range from CCTV camera-surveillance of suspicious behavior in public places to secret Internet monitoring and data-mining.

Such technologies raise ethical and legal issues (notably issues of privacy) that must be confronted against the background of the legal and ethical issues raised by counter-terrorism in general. Legal questions arise about counter-terrorism in general, because recent informal co-operation agreements between European heads of government may conflict with pre-existing legal commitments on the part of the same governments to safeguard freedom of association, free expression and privacy.

The goal of the DETECTER project is to identify human rights and other legal and moral standards that detection technologies in counter-terrorism must meet, while taking into account the effectiveness of these technologies as judged by law- enforcement bodies responsible for counter-terrorism, and other relevant authorities.

More information can be found at: http://www.detecter.bham.ac.uk/

EU-SEC II

Coordinating National Research Programmes and Policies on Security at Major Events in Europe

According to EU-SECII project, security at major events remains a top priority for host nations, attendees, participants and neighboring countries. Our task in this instance was to elevate the standard of analysis research to ensure airtight security during major events.

Therefore, EU-SEC II project aims to assist, through the harmonization of national security research policies, in the creation of a European House of Major Events. The driving force behind this initiative is the need for effective security policies supporting the efforts of major events organizers.

The core aspect of the project is the application of advanced managerial skills to cooperation in security. National and international entities require coordination in order to ensure that their information and policies do not overlap, while at the same time permitting them to work together as a unified system.

The project also aims to synchronize private security technology providers with national security practitioners, thus facilitating development and supply of the most effective security technology.

More information can be found at: http://lab.unicri.it/eusecII.html

ENISA

European Network and Information Security Agency

ENISA agency has been established to enhance the capability of the European Union, the EU Member States and business community to address and respond to network and information security problems. Today, ENISA is a centre of expertise for the EU Member States and european institutions in Network and Information Security, providing advice and recommendations on security- and trust-related matters.

Objectives:

- advising and assisting the Commission and the Member States on information security,

- collecting and analysing data on security incidents in Europe and emerging risks,
- promoting risk assessment and risk management methods to enhance capability to deal with information security threats,
- awareness-raising and co-operation between different actors in the information security field, notably by developing public / private partnerships with industry in this field.

Some of its reports include:

- "*Who-is-Who Directory on Network and Information Security (NIS)*" - contains information on NIS stakeholders, such as national and European authorities and NIS organisations, contact details, websites, and areas of responsabilities or activities[16].
- "*Report on the state of pan-European eIDM initiatives* " - contains information on the origins and scope of the ambitions for European eID interoperability, and looks specifically at how these are reflected in specific initiatives [17]
- "*Web 2.0 Security and Privacy*" - describes in detail these risks and others, based around a set of architectural patterns characterising the Web 2.0 paradigm shift. It then recommends a comprehensive set of initiatives in web standards and architecture, as well as policy actions [18]

More information can be found at: http://www.enisa.europa.eu/

5 Research areas

In this section we overview key areas of research that can add value to the definition and construction of NI2S3. We cover security of SOA and Web services, SOA reliability and dependability, SOA dependability in NEC context, and validation and compliance testing methods.

5.1 Security of SOA and Web Services

Distributed systems are increasingly built on the basis of SOA and in particular the Web Services. SOA allows building systems based on loosely coupled self-describing services, which can be dynamically linked. The basic format for data exchange standard is an XML. For a long time before standards were developed for Web Services, they had limited capabilities to secure transmission, to ensure data confidentiality, integrity, privacy etc. - in general to ensure the safety of passing data in the distributed system. Often, different manufacturers used their own software solutions to ensure security in SOA - not compatible with solutions of other suppliers. Currently, the OASIS consortium founded by powerful players in the software development like IBM, Microsoft, Oracle, Sun and others standardized the security issues in Web Services. Using standardized solutions enhances interoperability and contributes to the increasingly widespread use of security in SOA and thus opens SOA to be used in more critical applications in business and military areas.

5.1.1 Basic building blocks of Web Services Security

Web Services Security is built on the technology of digital signature with private/public keys and symmetric encryption. Current work in the W3C organization on XML Security [54] is carried on by Security Working Group.

Web Services standards are based on XML Signature and XML Encryption standards.

Figure 4 shows the dependencies of relevant standards for Web Services.

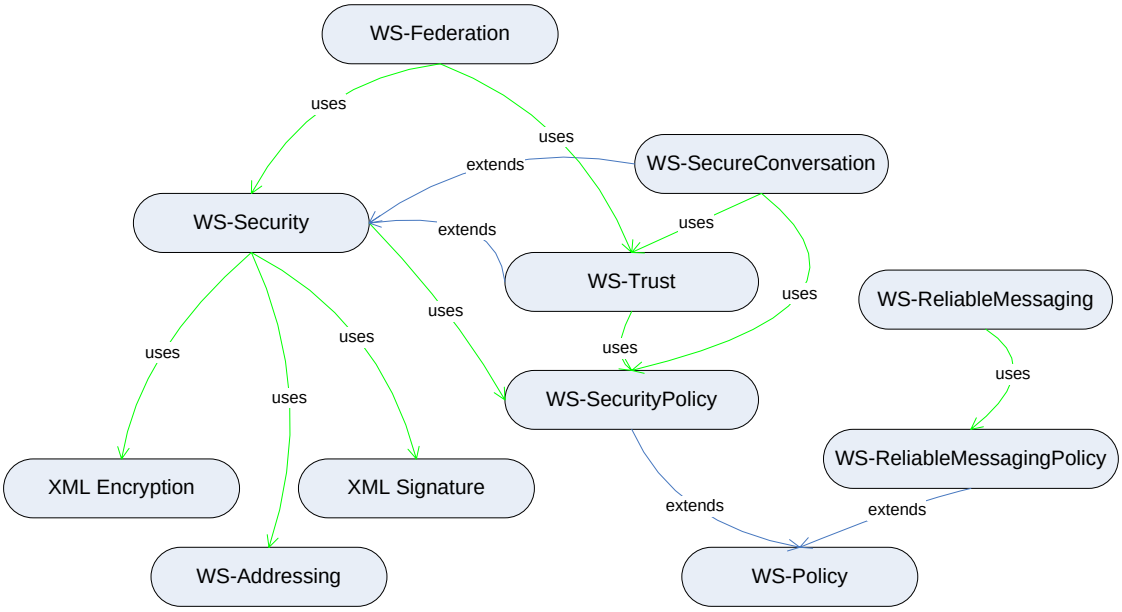

Figure 4 Web Services security related standards relations

XML Signature

This specification defines XML digital signature processing rules and syntax. XML Signatures provide integrity, message authentication, and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere [55]. As two XML elements can be logically identical they can differ in textual representation. Thus the canonicalization methods and procedures have been defined to obtain the same digest for logically identical XML elements[56] [57].

XML Encryption

XML Encryption is a specification that defines how to encrypt and decrypt data in XML. The data may be arbitrary data (including an XML document), an XML element, or XML element content. The result of encrypting data is an XML Encryption `EncryptedData` element which contains (via one of its children's content) or identifies (via a URI reference) the cipher data [58].

WS-Addressing

WS-Addressing provides transport-neutral mechanisms to address Web services and messages. This specification defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages. This specification enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner.

5.1.2 Web Services security standards

WS-Policy

WS-Policy is a baseline for other policies. Specific policies inherit from WS-Policy to extend it to particular needs. WS-Policy defines a framework for allowing web services to express their

constraints and requirements. Such constraints and requirements are expressed as policy assertions that usually are part of WSDL describing the web service [59].

WS-Security

WS-Security ensures that a message isn't tampered on route from the client to the server and that sensitive information (such as passwords) is encrypted. It defines a set of enhancements to the SOAP specification of messaging to enable protection of the message through authentication, confidentiality, and assurance of integrity.

WS-SecurityPolicy that is based on WS-Policy describes the security requirements and constraints of WS-Security enabled web service. Figure 5 presents a scenario with client connecting with WS-SecurityPolicy defined access method to a certain secured web service.

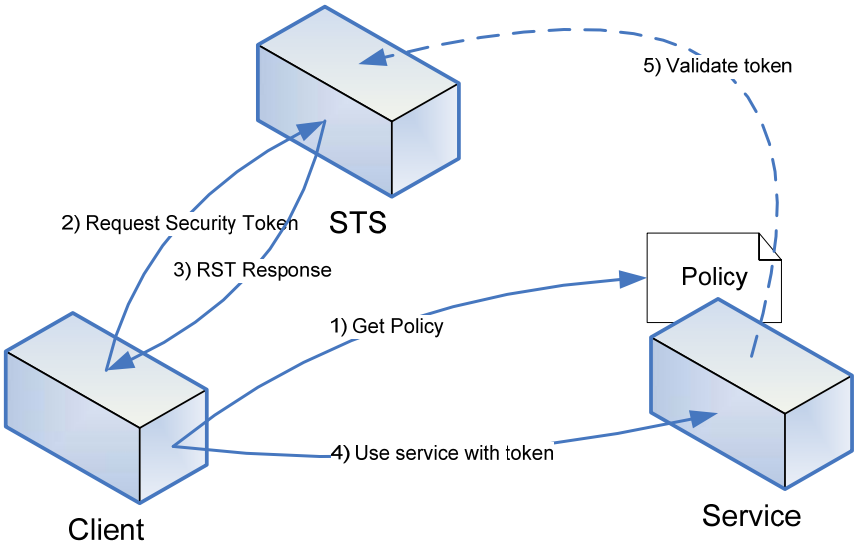

Figure 5 The role of WS-SecurityPolicy (Policy) in WS Security. Policy (1) describes that access to the service (4) requires authentication/authorization that can be achieved by acquiring token from SRS (2,3). Secured Service can validate token (5) by accessing SRS.

WS-SecureConversation

This is a set of enhancements to SOAP that allow specifying how a message can be secured throughout a long-running message exchange. WS-SecureConversation addresses the case when multiple messages are exchanged between nodes and it would be much more efficient if one could establish a context that reduces the overall burden of securing each message separately. The WS-SecureConversation specification defines a Security Context Token that is used in that conversation [60].

The security context is defined as a new WS-Security token type that is obtained using a binding of WS-Trust.

WS-Trust

This specification defines extensions that build on WS-Security to provide a framework for requesting and issuing security tokens, and to broker trust relationships. This specification uses these base mechanisms and defines additional primitives and extensions for security token exchange to enable the issuance and dissemination of credentials within different trust domains [61].

Figure 6 presents a scenario when a client from a separate domain accesses the secured web service in the other domain. The WS-SecurityPolicy defines the access method which is based on token issued by Security Token Service in the same domain as the service. The trust between domains allows the client to achieve a security token from its own SRS and then retrieve a token from service's domain SRS.

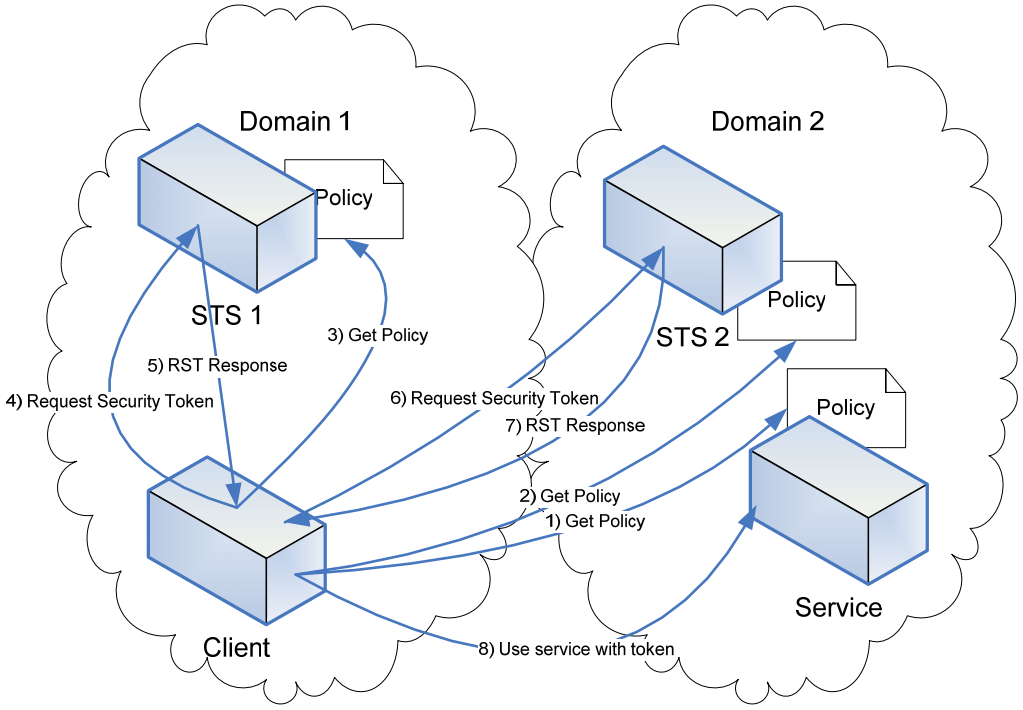

Figure 6 Security token provided by Security Token Service in Domain 1 (local to client) which is trusted for Domain 2, allows client to access Service in Domain 2

WS-Federation

This is a set of enhancements to SOAP that allows federating trust credentials among a group of Web service partners.

WS-Federation specification defines mechanisms to enable identity, account, attributes, authentication, and authorization federation across different trust realms. The WS-Security, WS-Trust, and WS-Policy models define the basis for federation. WS-Federation extends these

by describing how they may combine to enable richer trust across and between domains and federated services.

Although the final decision of access control is strictly controlled by the domain, which controls the resources, the federation provides mechanisms that allow the decision based on the choice (or mediation) of identity, attribute, authentication and authorization assertions between domains. Choice of mechanisms depends on the trust relationship between the domains. Generalized federation framework is able to integrate the existing infrastructure without new major investments in infrastructure. This means that the types of safety and infrastructure can vary in domains. WS-Security and WS-Trust specification allows for different types of security tokens, trust infrastructure and topology. WS-Federation uses these elements in order to identify additional mechanisms of the federation, which extend these specifications and leverage other WS-* specifications [62].

5.1.3 Other WS standards relevant for NI2S3

Apart from strictly defined standards for provision of security for Web Services there are some additional WS standards that may be particularly useful for Network Enabled Capability systems that target to provide fast and reliable building of services.

WS-ReliableMessaging

WS-ReliableMessaging is designed to ensure reliability of message interchange between distributed applications. Reliable message delivery means the ability to ensure that a message will be delivered with the desired and specified levels of quality of service. Some examples of this are:

- Message sent at least once (guaranteed delivery)
- Message sent at most once (guaranteed duplicate elimination)
- Message sent exactly once (guaranteed delivery and duplicate elimination)

WS-ReliableMessaging is designed to maintain reliability characteristics even in the presence of software component, system, or network failures. Thus it allows extending Web Services dependability. The protocol is transport-independent; allowing it to be implemented with network technologies other than SOAP, but a SOAP binding is also defined within the specification [63].

Nodes that are going to exchange data with WS-ReliableMessaging use WS-ReliableMessaging Policy assertions to indicate the required quality of service of the message sequence [64].

WS-Discovery

This specification defines a discovery protocol to locate services. In an ad hoc mode of operation, probes are sent to a multicast group, and target services that match return a response directly to the requester. To scale to a large number of endpoints and to extend the reach of the protocol, this protocol defines a managed mode of operation and a multicast suppression behaviour if a discovery proxy is available on the network. To minimize the need

for polling, target services that wish to be discovered send an announcement when they join and leave the network.[65]

The SOAP-over-UDP is necessary to WS-Discovery [66].

5.2 SOA reliability and dependability - open issues and challenges

Dependability addresses the key aspect of whether a system can effectively carry out its defined functions. It is important a system operates correctly in a dynamic environment of complex distributed systems and services in various organizations, where it is not possible to verify that they are trouble-free.

Dependability as applied to a computer system is defined by the IFIP 10.4 Working Group on Dependable Computing and Fault Tolerance as: "[..] the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers [..]".

An alternative and broader definition is provided by IEC IEV 191-02-03: "dependability (is) the collective term used to describe the availability performance and its influencing factors : reliability performance, maintainability performance and maintenance support performance".

It is worth noting that dependability is really a broad concept and should not be narrowed down to, for example reliability only. Reliability is only one of many attributes of general concept of dependability. It is better illustrated in Figure 7 which shows ontology of the dependability [67].

Similarly, security property of a system can be defined as a property that covers attributes like safety, integrity and confidentiality. In this sense dependability is a concept broader than security, which includes security (see Figure 7).

5.2.1 Dependability ontology

A number of factors affect dependability of a system [68]. Dependability can be characterized by the use of three concepts:

1) Attributes: ways to measure dependability of a system;
2) Threats: possible things that can affect dependability of a system;
3) Means: ways to increase dependability of a system.

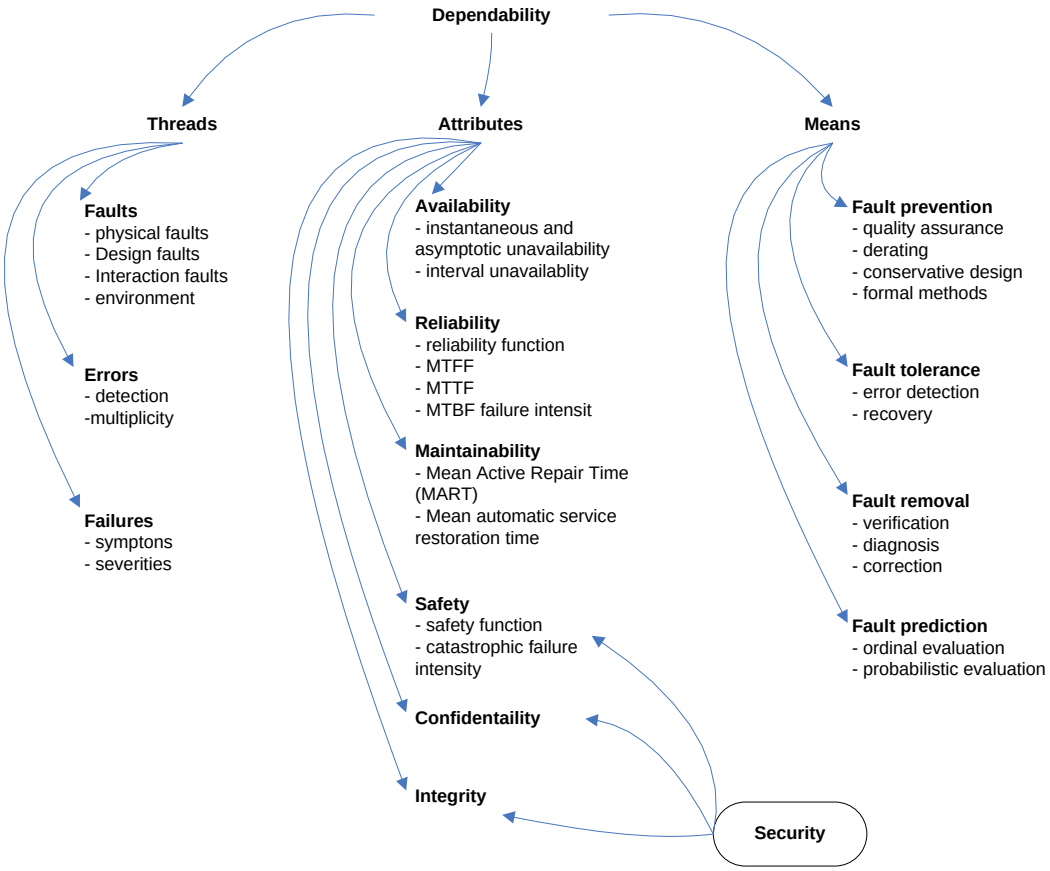

Figure 7 Structure of the vocabulary on dependability

5.2.2 Attributes of dependability

Attributes are measurable (quantitatively or qualitatively) properties of a system that allows determining its overall dependability.

1) Availability - the probability, that a service is present and ready for use – sufficient capacity is provided,
2) Reliability - the capability of maintaining the service and service quality (response times, delays etc),
3) Safety – the absence of catastrophic consequences,
4) Confidentiality - the absence of unauthorized disclosure of information - information is accessible only to those entities, who are authorized to access it;
5) Integrity - the absence of unauthorized or improper system alterations;
6) Maintainability – how easy handling of modifications and repairs is.

Some attributes like availability and reliability are quantitative and possible to measure while the rest are more relative or subjective like, for example, security. Measurements of dependability attributes can also become a significant QoS attributes for describing services

provided by a system. Some of the characteristics are also related to security, which deals with intentional threats to systems trustworthiness.

5.2.3 Threats to dependability

Threats are things that can affect a system and cause a drop in dependability. There are three main terms here:

- Fault: A fault (or a bug) is a defect in a system. The presence of a fault in a system may or may not lead to impairment or a failure. Although a system may contain a fault, its input and state conditions may never cause this fault to be executed so that an error does not occur and thus never exhibits as a failure.
- Error: An error is a deviation of actual behaviour of the system from the intended behaviour within the system boundary. Errors occur at runtime when some part of the system enters an unexpected state due to the activation of a fault. Since errors are generated from invalid states they are hard to observe without instrumentation (debugging, log files)
- Failure: A failure is a manifestation of an error outside of the system boundaries – it is violation of its specification. An error may not necessarily cause a failure, for instance an exception may be thrown by a system but this may be caught and handled using fault tolerance techniques so the overall operation of the system will conform to the specification.

It is important to note that Failures are recorded at the system boundary. They are basically Errors that have reached the system boundary and have become observable. Faults, Errors and Failures operate according to a mechanism called Fault-Error-Failure chain. As a general rule a fault, when activated, can lead to an error (which is an invalid state) and the invalid state generated by an error may lead to another error or a failure (which is an observable deviation from the specified behaviour at the system boundary).

Once a fault gets activated, an error is generated. An error may act in the same way as a fault in that it can create further error conditions, therefore an error may propagate multiple times within a system boundary without causing an observable failure. If an error propagates outside the system boundary a failure occurs. Since the output data from one service may be fed into another, a failure in one service may propagate into another service. Figure 8 illustrates the chain and shows the position of the means (fault prevention, fault tolerance, contingency and recovery planning) that helps improving the system dependability and deal with consequences of failures.

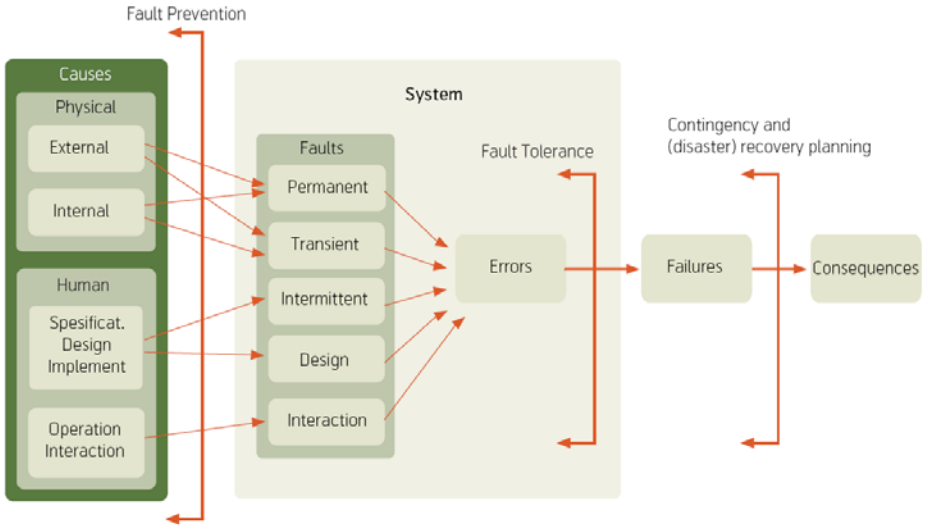

Figure 8 Fault-Error-Failure-Consequences chain and means to deal with them

5.2.4 Means to preserve dependability

Means to increase the dependability of a system is to break chains of the mechanism of a Fault-Error-Chain. Four means have been identified so far:

- Prevention
- Removal
- Prediction
- Tolerance

Fault Prevention deals with preventing faults being incorporated into a system. This can be accomplished by use of development methodologies and good implementation techniques.

Fault Removal can be sub-divided into two sub-categories: Removal During development and Removal During Use.

Removal during development requires verification so that faults can be detected and removed before a system is put into production. Once systems have been put into production a system is needed to record failures and remove them via a maintenance cycle.

Fault Prediction forecasts likely faults so that they can be removed or their effects can be circumvented.

Fault Tolerance deals with putting mechanisms in place that will allow a system to still deliver the required service in the presence of faults, although that service may be at a degraded level.

Dependability means are intended to reduce the number of failures presented to the user of a system. Failures are traditionally recorded over time and it is useful to understand how their frequency is measured so that the effectiveness of means can be assessed.

Dependability attributes in the systems based on autonomous services

In distributed systems, where the functions are provided based on services from different participants that may be autonomous, new challenges appear to dependability.

Availability - A participant within a business process may have no control over the availability of services provided by other participants; service availability may be dynamic and unpredictable.

Reliability - A participant within a business process may have little or no knowledge of the reliability (including the performance) of services provided by other participants; this is especially a problem in long running interactions

Confidentiality - A participant within a business process may have little or no knowledge of the access procedures in place to protect the confidentiality of data shared with services provided by another participant.

Integrity - A participant within a business process may have little or no knowledge of the security procedures in place to protect the integrity of services provided by another participant.

The SOA Web Services Security standards like WS-Security and WS-Trust and WS-Federation partially addresses the problem of confidentiality and integrity attributes for systems build on autonomous services.

In such environment the critical is to maintain information upon the services that may be troublesome. This issue may be addressed by accountability extensions to support dependability.

5.2.5 Accountability extensions to dependability

When users invoke services in their business processes, they expect them to produce good results that have both functionally correct output and acceptable performance levels in accordance with quality-of-service (QoS) constraints, such as those in service-level agreements (SLA). Service level agreements (SLAs) are formal contracts negotiated with the customers. SLA definition typically includes SLA templates, which are predefined contract templates, QoS parameters defining what can be measured for a particular service (either business or technical perspective), Service Level Objectives (SLOs) are particular values for a given SLA on QoS parameters. Alternatively to SLOs, QoS Classes of Service, which bundle different parameters and their thresholds to provide standard choices for the customer [69]. The Quality of Service for a service can be specified using the Web Service Level Agreements (WSLA), a language from IBM [70]. So, if a service produces incorrect results or violates an SLA, an enterprise must hold the service provider responsible (also known as accountability).

Identifying the source of a business process failure in a SOA system can be difficult, however. For one thing, business process can be very complex, having many execution branches and invoking services from various participants. Moreover, a service's failure could result from some undesirable behavior by its predecessors in the workflow, its execution platform, or even its users. To identify a problem's source, an enterprise must continuously monitor, aggregate, and analyze business process services' behavior. Analyzing such a massive amount of information requires efficient support from that enterprise's service-deployment infrastructure. Moreover,

the infrastructure should also detect different types of faults and support corresponding management algorithms.

In general accountability can be defined as “the availability and integrity of the identity of the person who performed an operation.” Both legal and financial communities use the notion of accountability to clarify, who is responsible for causing problems in complex interactions among different parties. It’s a comprehensive quality assessment to ensure that someone or something is held responsible for undesirable effects or results during an interaction. Accountability is also an important concept in SOA, because all services should be effectively regulated for their correct executions in a business process. The root cause of any execution failure should be clearly inspected, identified, and removed to control damage. The accountability is the main target of LLAMA Project.[71]

5.3 Challenges and open issues for SOA dependability in NEC context

From NEC perspective the dependability of Service Oriented Architecture can be challenged in following areas:

- Service integration - Dynamically composable services to achieve critical capabilities. Problems of granularity of services and its partitioning with relation to hardware resources. Recognizing the abstract functions that current and future platforms provide will be key in defining services at an appropriate level to interchange resources while maintaining a dependable system. At the capability level, abstract service definitions of functionality and QoS attributes can be combined to achieve a mission objective, independent of the service implementations.
- Service discovery - Identify service types to define integration.
- Enable dynamic binding during operations
- Service reconfiguration - Adaptation of service and service integration to meet on-demand requirements. Loose coupling allows resources to be changed to meet new challenges – the problem is what an impact on dependable operation is.
- Runtime monitoring and management (instrumentation)
- Service evolution - Adaptation of implementations to meet changing consumer needs

5.3.1 Fault tolerance in SOA

Building reliable composite systems from unreliable services **i**s an **i**mportant aspect of systems integration**.** FT-SOAP [72] follows the service approach in fault tolerant CORBA and provides transparent fault tolerance by upgrading SOAP with additional components to support fault detection and replication management. FT-SOAP extends WSDL to inform the clients of the replica information. Other approaches to fault tolerant services concentrate on centralized service composition or orchestration. Proposed techniques include using BPEL compensation [73] and fault handlers to achieve fault tolerant composition[74].

5.3.2 Replication

One of the means to increase availability and fault tolerance of SOA systems are achieved by service [75],[76],[77] replication. Trivial solution of replication introduces problems like expensive communication when all-to-all request/response is sent. Proposed solution includes coordinator-follower approach where a dispatcher acts as a proxy for client requests, and sends them to service replicas in parallel [78]. The other issue with replication arises from replication of data. The optimal solution is to have each node storing a share of data and coordinated computation is organized with multiparty computation algorithm [79].

5.3.3 Middleware extensions

Various middleware have been proposed to improve fault tolerance. The solution [80] is based on an introduced abstract concept of Service Plan containing the whole business process with defined tasks, structure of process and set of the settings and attributes of the process. This Service Plan is run on QoS aware middleware platform. Tasks can be accomplished by many candidate services provided by the Service Communities (design diversity) and in any time given service from one community may be replaced by another from a different community service. Basically, this solution is based on intercepting requests and routing them to selected services in order to optimize the parameters of QoS and reliability of the business process.

5.3.4 Real time SOA

Real-time SOA is the new paradigm for building next-generation, real-time infrastructures and devices under service-oriented computing. However, current SOA solutions have not addressed the strict predictability demands that many enterprise applications require, from banking and finance to industrial automation and manufacturing.[81]

To support RT-SOA also building blocks of whole system must support real-time requirements:

- Operating System. Any real-time middleware framework must be built atop an operating system that provides real-time scheduling and a fully pre-emptive kernel.
- Communications Infrastructure. The communications infrastructure must provide predictability guaranteed by QoS in the network [82]
- Business process composition Infrastructure. The SOA composition infrastructure must be able to generate a business process that satisfies both a user's functional and timeliness requirements. It must be able to negotiate such timeliness requirements with distribution middleware to ensure predictable business process execution [83]
- Distribution Middleware. The main purpose of a real-time middleware platform (like a real-time enterprise service bus (ESB)) is to ensure the predictable execution of individual service requests [84]. Therefore it is essential to provide support for advance reservations and avoid overloading or overbooking of a given host's resources.
- Client Infrastructure. Many existing SOA deployments use a business process execution language (BPEL [85]) engine, which is a centralized mechanism to coordinate all remote service interactions within the process. An RT-SOA solution must address any unpredictability that a BPEL engine may produce.

5.3.5 Evolving architecture

For NEC uses, the very important issue is the ability to adapt architecture to the changing conditions of the work environment. Loose-coupling of SOA allows flexible changes and upgrades of components that build the system, but the important issue is that the architecture allows such changes - because unpredicted changes made in the future, usually results in loss of dependability attributes and increased probability of errors. [86]

5.3.6 Evaluation of architectures

In order to evaluate the architecture it is essential to define metrics and measures. Basically metrics rely on the attributes defined in dependability ontology like availability, reliability, safety. The current work should focus on the definition of measures and practical procedures on obtaining such measures. Work is this area is carried out in NECTISE project [87]

5.3.7 Instrumentation (Tracking and Monitoring)

Crucial means to evaluate dependability is the quality assurance which basically can be guaranteed by constant monitoring. The monitoring is implemented usually as instrumentation in platforms of services' containers. The typical problem with monitoring and tracking is that it needs to manage huge amounts of data so the approach needs to be adaptive to conserve resources.

5.4 Validation and Compliance testing methods

In all software there are bugs, also inside the ones we usually use without experiencing any problems. In these years there is an increase of the connectivity demand, all the software has to access the Internet, with relevant risks coming from bugs that could be potentially exploited by an attacker to compromise security services like privacy, integrity or data availability. Therefore it's really important to test software, especially the one designed to use a network connection.

This section is a survey on the state of art of black-box testing for software, with focus on security testing for network software. We also consider automatic tools to help testing process and we analyze some tools used by tester communities.

5.4.1 Testing

The analysis process of a software system, intended as a comparison between the real behaviour with the attended one, is defined by *IEEE Standard Glossary of software engineering terminology* with the word *testing.* When we evaluate a software system, we try to understand if it complies with specs, otherwise if it has some properties like security. IEEE defines the terms mistake, fault and failure: a developer can make a *mistake* writing code; while software running, this can cause a *fault*, for example an erroneous command or definition or expression: this is usually called "error" or "bug". The presence of this fault can cause a *failure*, in other words the software enters in a state where it can't perform the wanted functions.

The base unit in testing process is the *test case*, that focus on some aspects or on evaluation

of design features. Many test cases are reunited in a *test group*, and they are reunited in a *test suite*. Every test case has an aim (*test purpose*) and a test group can have a common aim (*test group objective*). The test result is the *verdict* and it can be: *pass*, *fail* or *inconclusive*. A test case with a pass verdict means the software meets the aim of this case and the tested code has a correct behaviour. A fail verdict means that there are violations of design specs, whereas a case can be inconclusive when it isn't possible to give a pass or fail verdict.

Black box, white box and grey box testing

Software testing can be classified by the method used, it can be:

- Functional testing
- Structural testing

Functional testing sees the item under test like a black box: an object that takes some input and responds with an output, that we can evaluate the response using the design specs. Using this method, the tester is like a normal user, he doesn't know anything about implementation details, and he works comparing the expected features with the real behaviour of that system. For example black box testing can be used with either design specs, or protocol definition or interface description as input.

In structural testing, a tester uses knowledge about software implementation like source code details, so it is called white box testing. This method of testing can be also classified in dynamic or static analysis depending on how the software is tested; in the first case a program is tested when it is running, in the second when it isn't.

Between black and white box testing an intermediate method can be defined. It is called grey box testing: in this case it uses a limited knowledge of software internals; in other words the tester knows only some part of code, not all, or usually he has access to some design specs more detailed than the original system requirements. For example a test case can be generated on system's architecture diagram or states model.

In grey box testing, like in black box case, software is tested from outside by modifying input. Test case creation, however, is different because it is based on a deeper knowledge about the tested system.

Testing level

Usually testing process can be divided in different levels:

- Unit testing: a unit is the smallest testable part of the system, in other words the smallest part of code that can be compiled, linked and loaded in memory or used under control of a driver. A unit is the work of only one developer and can be several hundreds of lines of code.
- Component testing: it is executed on a group of different unit.
- Integration testing: integration is the process where different components are united to form a more complex component; it is useful to test also bigger components even if tests of single part are OK.

- System testing: a system is a complex component. This testing level is aimed to find a bug that is neither inside a single component nor related to the interaction among them. System testing focuses on behaviour or problem that can be found only on the whole system or majority of it. This testing includes analysis of performances, security, start-up process and recovery.

Conformance testing

OSI Conformance Testing Methodology and Framework (CTMF) defines a methodology for conformance testing of protocol implementation: the goal is to verify if the implementation behaviour accomplishes compliance requisites defined by a protocol. This testing uses a functional approach and the same test cases can be used on different implementations of the same protocol.

5.4.2 Software security testing

Many security violations are caused by software bugs. These vulnerabilities can be generated both in the design phase and in the development phase. Design vulnerability comes from system planning errors; Implementation vulnerabilities are generated by a developer while coding and they are derived from bugs in source code: even if system design is without errors and it uses protocols or algorithms, theoretically guaranteeing a strong security level, implementation flaws can still make a software extremely vulnerable. During the design phase a possible option is to use a formal model that prove security features while during the implementation phase the developing complexity may often overcome any formal test capability.

Exploitation of Implementation vulnerability

Implementation vulnerabilities are generated from errors made by developers, and they are different from common code bugs. Implementation flaws for example can affect correct software behaviour in a way that is visible also to the users. On the opposite, a vulnerability can be used by an attacker to get super-user permissions or to compromise system features. Usually vulnerabilities are not spotted directly by the users and they're likely to sit in the dark waiting to be exploited. Therefore it's very important to run tests that exercise as many states of operation as possible to eliminate all the vulnerabilities of a particular implementation in order to prevent potentially bad software usages. A software or method that uses a vulnerability to compromise a particular system is called an exploit. This type of attack has different results ranging from both denial of service to complete system violations. When this happens only a patch can solve the problem, resulting in a very expensive process. The patch code has to be developed, properly tested and applied to target software to remove the vulnerability causing the exploit. A lot of systems can still remain vulnerable even after the patch has been released, especially embedded systems which are more likely to lack an upgrade feature.

There are many kinds of vulnerabilities. Common errors are using an input without a validation, using insecure library functions or using libraries in an insecure manner. Programming languages like C or C++ can be problematic from a security point of view because it's really

simple to unwittingly write insecure code. For instance a very common and serious vulnerability is *buffer overflow*. An error in a memory control that occurs when a process writes some data without checking if a large enough memory space is assigned to it: if the space is not large enough a part of information is written out its memory space and it overwrites other data; this kind of bug usually cause a crash, but with properly chosen input may permit to an attacker to run arbitrary code with the same permissions of victim process.

Robustness testing

A very important feature for software granting security services access is its robustness. A robust software has to tolerate a dysfunctional formatted input possibly chosen by an attacker to cause a behaviour that isn't wanted by design. Robustness problems are security too: several vulnerabilities can be exploited to compromise a system. This is a really serious problem for network software, especially for public Internet services which are naturally exposed to wider risks. Today a great number of common software has robustness problems that can be exploited to cause security vulnerabilities. There are several services on the internet which track known robustness flaws and hence vulnerabilities.

Robustness testing is different from traditional conformance testing, since conformance testing focuses on verifying if tested implementation has a correct behaviour according to protocol definition. On the contrary robustness testing focuses on software capability of handling and tolerating dysfunctional events and attacks. This kind of testing looks for implementation vulnerabilities.

In the opposite of conformance testing, it doesn't check the system output, but it evaluates if software behaviour is secure and robust in presence of malformed inputs, also seeking if failures occur.

The main difference is that conformance testing uses inputs that are expected for the particular software or protocol, while in robustness testing the inputs are unexpected. The latter is also different for the number of test cases usually written. Robustness testing can be thousands more tests over and above conformance testing.

Protocol testing

Protocol implementation is a consistent target for security analysis: messages are transmitted over the Internet or over other insecure network, and they are exposed to different type of attacks; for example any type of cryptographic protection is useless if an attacker is able to handle a valid session.

5.4.3 Technology for black box testing

This section lists some technologies used in black box testing with focus on implementation vulnerabilities. We take in consideration that different methodologies don't exclude each other, but they have overlapping features. Also, when we consider a test process, often we merge different technologies.

Fuzzing

Fuzzing is the name of a utility created by Miller: it's a random characters generator that can be used for testing and it sends random data to the interface of the testing system. With fuzzing we mean introducing noise in a software interface. For example a fuzzer can intercept a system call of a software component that reads a file and can substitute the file content with random chars. Therefore fuzzing focuses on finding abnormal behaviour through the insertion of noise in the input. Abnormal behaviour can be a symptom for presence of bugs.

Fuzzing methodology is evolved and now overlaps many testing typology like domain testing, syntax testing, exploratory testing and fault injection.

Shortly fuzzing is a blind search of unexpected bug in software. For example, above we discussed how a tester replaces a file with random data read by a software under test; if eventually the software crashes, the tester can prove that the application doesn't check its input and that it considers all the input files as properly encoded. This lack of control can be exploited by an attacker that change the input file content to attack the software and afterwards to obtain process permissions. For many interface a fuzzing approach is too simple or inadequate. For example if we want to test a web server with a CMS, fuzzing will hardly create valid URLs, so they'll be immediately rejected by the URL parsing algorithm. In this case we test that component but we cannot reach any other part of the program. So a completely random fuzzing approach is often useless for bug hunting.

Nowadays this technique is evolved and it uses a smarter approach: for example a fuzzing tool knows common Internet protocols, so it is possible to select which part of data have to be modified and randomized. Also they can select from a list of values the data to be inserted: a tester can make a range of input to test without using really random inputs. In this case we have an overlap between fuzzing and syntax testing.

The set of fuzzer inputs is made by all possible combination of data input for the tested software, so it is virtually unlimited: to limit it, often we use some heuristic depending on our testing focus. For example, if we want find a buffer overflow it's useful to send long data stream. Thus, test cases are generated using sets of inputs and a heuristic. Fuzzer software can be divided into two groups, depending on the test case generations:

- Generation fuzzer (also known full-blown fuzzer) is autonomous software that create almost valid session: it can build input data from a model, that can be a base model, like random data, or more complex like a model of the tested protocol.
- Mutation fuzzer (also known capture-replay fuzzer) takes a known valid session and alters data so it can create almost valid sessions.

A Mutation fuzzer usually has a general purpose, while a generation fuzzer is often specific for a single protocol or application.

We can describe a fuzzing test in three phases:

- Tokenization. The protocol is split in variable and not variable components, the second ones aren't fuzzed and they are header or constant; the first ones are instead changed by fuzzing and they can be strings integer or binary data.
- Traffic Generation. The fuzzer generates network traffic towards the tested component. It's the most sensitive step because if it's done in erroneous manner, most part of sent data will be ignored.
- Traffic alteration. Traffic, created in the second step, is changed inserting random data so there are more chances that a bug happens. This step can be done simply with a bit flipping or with more complex approach. When the fuzzer alters some variable data it's also probably needed to change other complementary part of the traffic, like the field length, to keep the session coherency.

Domain testing and Syntax testing

When a test case uses input specs we talk about domain testing. Possible input values are grouped in input domains, group of values that are managed in same manner, because software components use these values in similar ways. Test cases have to look for domain boundary because there is more probability they are erroneously managed by software. The aim for this kind of testing is to check the code behaviour in presence of a valid input.

Syntax testing is another approach to generate test cases based on input specs, they aim to understand what happens when input data breaks protocol syntax: we can test using random data, with an element in random order or left out, and so on. A common test case is a long input looking for buffer overflow. So syntax testing helps to understand if input is filtered and handled correctly. Also valid inputs are useful to test different steps in the protocol without an immediate discharge of data like in random fuzzing.

Software interfaces can be different: command prompt, input file, shell variable, pipe, socket, and so on. An interface has a language that discriminates whether an input is valid or not; this language can be either open or hidden. While an open language can be a widely known network protocol an hidden language is something usually undisclosed to the final user; for example it can be a data structure definition inside C code used to assure communications between different software components.

For an automatic syntax testing a formal description of the tested language is needed. If the language is hidden, the tester has to create a specs, that describes it. To accomplish this task testers usually use BNF (Backus-Naur Form) and regular expression. Both can define a free context language: a language expression is valid if match with that definition. Once the language specifications have been defined it is used in syntax testing to create a base valid expression and then sent to the software to evaluate replies.

Test case creation starts with the insertion of a single alteration (only one point in the expression is not matching the free context language). After this step, tester can use multiple alterations; numbers of test cases exponentially increase with the number of errors introduced in the test.

There are different kinds of errors of alteration that can be used:

- Syntax error: error in the language grammar, they are created when a field is deleted or a different field is inserted or more field are sent in erroneous order.
- Delimiter error: delimiter define separator between fields; in ASCII coded language fields are characters and delimiter is space, tab or line-feed or other symbol like dot or comma. If tester want to insert this type of error can remove a delimiter, or he can insert it more times or change it with another character.
- Field value error: usually a field has a valid range of values: tester insert an error using a value that is out of this range.
- Context dependent error: error in some characteristics that cannot be described by a free context language.
- State dependency error: a valid expression usually cannot be accepted in every internal state of software, so an error of this type can be the sending of valid expression when software isn't in a correct state.

Usually valid sequences of data can be created automatically allowing the tester to focus on the creation of test cases and to run a huge number of different tests. In syntax testing the tester can automate the testing process developing a driver specific for the tested application. Testing tools may support him during the test suite creation.

A testing tool focused on security implements common protocols and input formats because software communicates to each other on private or public networks. So a security tool can be used for protocols such as HTTP, FTP, SMTP, SQL, LDAP, SOAP and XML.

Many attacks use a data injection approach sending look-alike valid data sequences with hidden malicious data. If not checked or validated correctly, an input coming from a user can be exploited to add malicious instructions that can result in a security violation; for example a data transferred by a URI can be used in a SQL query to create an SQL-Injection attack.

Exploratory testing

When we want to perform security testing, one option is when we don't have any clue about the expected reply of the tested item. So a tester can continue the testing process using replies to create the following test cases. When the tester explores the software behaviour, he takes information about how the code works to continue testing. Many black box approaches can be used also for this method, but some are closer than others; for example fuzzing, because often a tester hasn't any idea what he's looking for and so he sends random data inside software interface.
So testers aren't aware of the kind of errors they are going to spot, they're just looking for software dysfunctional behaviour. Some alternative approaches, which can be catalogued like exploratory, are stress testing and fault injection.

Penetration testing

A particular exploratory approach is penetration testing: it is a vulnerability research based on a real attack to the system. Usually it is performed by a group of people called *tiger team*. During this process a tester try to find old vulnerability found in similar systems. This test can be done manually or automatically with tools called security scanners; this testing is strongly based on tester experience.

Stress testing

Stress testing proceeds by submitting a huge amount of input to the tested system; for example a tester can send the same data many times or a big size input. This approach includes also the creation of an extreme condition, like full memory or hardware failure. Usually the aim is to guarantee quality of service in difficult situation; but considering a security point of view, it's useful to search other anomalies. For example a particular condition can cause the software to call some error handling routine, these routines can have vulnerabilities and only with this approach a tester can found that kind of bugs.

Obviously an attacker can create any particular condition in order to be successful in his attack.

Fault injection

A fault injection technique is usually applied to hardware testing, where some parts are voluntarily damaged, and tester evaluates system robustness. In software fault injection we change the code, or input, and we monitor system behaviour. So we see how software resists in particular condition, and we evaluate its robustness. If the tester changes the software code he's performing what we call white box testing; sometimes the tester can step in ongoing communications between different running components, in this case we can talk about black box testing.

Fault injection is useful for stress testing, but it can be used to create the same condition experienced by an attacker; for example tester can change some data in system calls between a process and operating system so to simulate an attacker taking control of some external resources.

Often we cannot understand how a bug can be exploited to compromise a system. The real and valuable meaning of the whole testing process is to find those bugs so they can be removed.

Data analysis

Data analysis is a process whose main goal is to analyze the information contained in the output of a given component. This is more useful than the simple observation, and the tester can change the behaviour of an application. Security testing uses this approach to understand if an attacker can perform alerting operation on the system. Two important things about this kind of testing are:

- Stateless protocol use external methods to keep state of transmission (for example HTTP use cookies), so it's dangerous to let an attacker access that information; data analysis can help to understand which information can be stolen.

- Some cryptographic functions use random number generator, for example when it's needed to generate secure key. If an attacker can collect a lot of output from a random number generator and that generator has a bug, the attacker is then able to predict future output. Therefore a strong secure algorithm can be exploited too. The use of a weak random number generator is a design error, but testing is useful to search if code implements correctly the design, or to analyze third part code.

Software monitoring

Monitoring software behaviour is a really important part of the testing process, because it can help stating the test case result. Monitoring a software requires to collect information about different aspects of the tested system such as memory usage, cpu usage or network replies. Once those data are stored it is necessary to evaluate if there is an error or not. In security testing this process is more difficult because tester doesn't compare the software behaviour with an expected one: tester looks for evidences of vulnerabilities presence. Also it's important to monitor all system, like memory use or network transmission, also the one that are less visible than a blue screen of death!

There are many tools for monitoring automation, and they are helpful for those approaches requiring to send and analyse a large amount of data like fuzzing.

5.4.4 Known tools and software

There are many tools to help a tester in his work, and they differentiate for generality and automation. Usually these tools work on only one protocol and use a fuzzing approach. Another kind of help is given by some libraries or frameworks, which help testers to develop their own tools. Also there are frameworks that support testing process with organisation of activity. The section below provides a survey of the more relevant tools used for testing process.

PROTOS and PROTOS protocol genome projects

Classic PROTOS

Project *Security Testing of Protocol Implementations* (PROTOS) start in 1999, inside University of Oulu and VTT Electronics, and it focuses on finding different approaches for black box testing of protocol implementation. Testing evaluates robustness of software that implements high level protocol like HTTP or SNMP, and they define a method called: mini-simulation method. The main goal of this approach is the creation of a great number of messages with only one faulting element, and otherwise correct in all other parts.

This method has the following steps:

- We take the specs of interface that we want test, and then we use BNF for defining a free context grammars.
- We create a few test cases, which are valid communications for that protocol: these cases validate BNF protocol model and give an evaluation of transmission with tested software.
- Using BNF we create manually a set of anomalies, or we can reuse an old set.

- We add the set of anomalies in the valid case.
- Valid elements and anomalies are used to create many test cases. It uses all combination of errors and all possible error values, and for each combination it designs a test case.
- Test cases are automatically created: it makes binary PDU, BNF description and documentation. After this step if someone wants to add an anomaly or a value he needs to re-run all processes and to re-build all test cases.
- Test cases are used for testing. Stateless protocol PDUs are sent to the tested system, for stateful we must evaluate BNF to reach the state that have to be tested.
- We analyze the log for test results.

This method is used for robustness testing. Subsequently it has been used to validate the protocol implementation. PROTOS works on client-server software that implements request-response protocol like LDAP, SNMP, SIP and H.323.

PROTOS Protocol Genome Project

PROTOS Protocol Genome Project started in 2003 from PROTOS Classic; the goal is to create a system that automatically design faults, error and so test cases starting from valid data. This approach, called *model inference assisted fuzzing*, uses a set of data for training and automatically designs a model upon which all test cases for robustness testing are built:

- We collect a training data set, examples of valid data about tested protocol.
- A model is created.
- Model creates data look-alike original ones.
- Those data are used to test and to search abnormal behaviour.

This approach is used to test for software viruses. It makes a large number of archive files, modified by the model, and then several anti-viruses scanned by them.

The quality of model depends on training data sets: if examples aren't complete, some features will not be tested. The main flaw of this approach is the lack of knowledge about domain, as is the semantic meaning of data.

Codenomicon Defensic

A spin-off of PROTOS creates Defensic, a commercial tool produced and distributed by Codenomicon. This software has some tools, each of which works on a specific protocol.

Fuzzer tools

There are many tools that implement a fuzzing approach for a wide variety of network protocols and common file formats; these fuzzers can be used with all software that implements a protocol or uses a file with that format.

Other fuzzers use a more generic approach so they can be used with many kind of protocols: they make simple mutation, that doesn't take into account syntax like bit flipping or byte transcoding.

Finally there are fuzzing frameworks which give more flexibility and are used to test unknown protocol (proprietary) or never tested protocol. Usually a good fuzzing framework has to help in some operation:

- Help in the creation of the model of protocol: some have tool that convert network traffic in a model;
- rebuild automatically the length of a packet: every times it modifies a field in a frame, it should calculate the new length of the packet and change length field to prevent immediately discarding;
- calculate CRC or other type of checksum and change it accordingly;
- give functions to generate pseudorandom data;
- include a set of data that are already successfully used in discovery a fault;
- give help to discover a fault: often a fault is spotted because the software does not reply anymore; more advanced techniques use a debugger;
- give a tool for reusing the code developed by the tester: so the fuzzer can evolve and become smarter.

There are many fuzzers used today. In the following we give a survey about them.

Spike

SPIKE was started in 2000: developed by Dave Aitel, Spike is a fuzzing framework. It's written in C and it provides some useful API to develop fuzzers for network protocols. It's an open source projects licensed under GPL and it includes common functions used to test a wide variety of both well known and specific protocols. SPIKE uses a fuzzing approach called *block-based*: protocol syntax is divided in blocks and these blocks are changed pseudo randomly. The blocks are defined with a structure (called spike), which contains binary data of a given length. Spike can be used to spot SQL injection, buffer overflow or format string bug.

The project is poorly documented and it isn't well organised, also a little change in framework, like an insertion of a new fuzz string, make it needed to re-compile all packages. Also SPIKE hasn't any tool for re-using code. In 2006 it was re-written in Python and now it's part of the CANVAS framework.

Peach

Written by Micheal Eddington, distributed by IOACTIVE under MIT license, Peach born in 2004 and it is a cross platform framework developed in Python. With Peach, tester can define a structure that have to be fuzzed using XML, creating a peach pit file; subsequently peach engine mutate data.

Framework can model:

- information about type;
- relationship like length and counter;
- checksum and other type of data validation;
- static data transformation (like zip compression);
- base model of finite state machine;

Peach can do different kind of fuzzing: both generation fuzzing (from structure Peach can create data) and mutation fuzzing (Peach can change data if a proper input is given).

GPF - General Purpose Fuzzer

It is a open source tool developed by Jared DeMott for UNIX system. It is designed like a generic fuzzer and it can be used in different ways: a tester can do random fuzzing, or can mutate data or can manually model a protocol with C written file called tokAids. This framework is really hard to use, his learning curve is very steep, and modelling a proprietary protocol is an hard task.

From 2007 GPF is used in Evolutionary Fuzzing System (EFS): it uses generic algorithms to generate the input that will be sent to tested software. EFS sends semi valid session to target application, that are monitored by a debugger. It stores the results inside a database and those results are eventually used by an algorithm to evolve the test case generation system.

Autodafé

Autodafé is a tool written by Martin Vuagnouxand and it is released under GPL license on UNIX systems. It uses an approach like SPIKE, based on block structure. This framework aims to reduce the degree of freedom of fuzzing. All test cases are relative to a protocol area that is more easily vulnerable. It uses a technique called *Marker Technique*, that mark with a label every variable fields of the protocol: a marker is a data (string or numeric) that can be changed by user or tester. For this reason Autodafé makes test processing variable with order defined by label, therefore it focuses on those variables that are more likely to cause vulnerability. Autodafé has a debugger that monitors the dangerous API (for example: strcpy() or fprintf()) that can be used by fuzzed variable. The marker system gives more importance to the label linked to these variables so they will be tested before. The framework will ignore less dangerous variables, especially the ones used by safe APIs, so space of input will be reduced.

Sulley

Sulley is a framework dedicated to fuzzer development constituted by a large number of easily upgradeable components. Authors say that Sulley differs from all the other fuzzers since it is not only focused on input data generation but also on data transmission and target monitoring.

Stateful protocol Fuzzing

Almost all fuzzing tools, also the ones that create a model of the tested protocol, don't consider the internal state of the target: they send data to the target without being aware of the state of the target software making difficult to choose an input required to reach a certain state. Consequently a fuzzer usually tests external protocol exchanges of data and it cannot check all possible transmissions. Only Sulley developers try to solve this problem, using a graph to monitor the internal FSM behaviour of the target.

6 Summary and Conclusion

This document presents the state of the Art in NEC and SOA and studies the envisaged use of SOA for the benefit of NEC. An initial analysis is made on key gaps with respect to security provisions for SOA in the NEC context. Previous and current projects are discussed with their relevance to SOA and NEC and general NI2S3 concepts.

It is clear through the reference material in this document that the state of the art in SOA itself is fairly well advanced with several examples of commercial use of SOA based infrastructure deployments. The research field is very active with regard to SOA with activity generally focused toward specific aspects such as security, dependability and compliance. Many relevant activities have been identified in this document which can form major input basis to the work of NI2S3.

With regard to NEC there is material mainly published through larger national defense frameworks which are primarily encapsulated within Enterprise Architecture Frameworks such as MoDAF, DoDAF and NAF.

No detailed material is found which uses NEC for the use by CIIP [3], although SOA is an enabling architecture. The Enterprise Architecture Frameworks do, however, present a broad application onto architectures which are not necessarily military based, although no material has been identified which deals explicitly with the concept of using SOA for NEC in the context of CIIP. This issue is expanded in the companion deliverable D1.2.

7 References

[1] The American Heritage Dictionary of the English Language, Fourth Edition. Houghton Mifflin Company, 2006.

[2] The European Programme for Critical Infrastructure Protection (EPCIP) http://europa.eu/rapid/pressReleasesAction.do?reference=MEMO/06/477&format=HTML&aged=0&language=EN

[3] E.M. Brunner and M. Suter, "International CIIP Handbook 2008/2009", Center for Security Studies, ETZ Zurich, 2008

[4] G Brunette et al, "Toward Systemically Secure IT Architectures", June 14, 2005 at WET ICE 2005, Sweden

http://www.sun.com/software/security/docs/systemic-security-wp-1.pdf

[5] UK Cyber Security Strategy Launched http://news.bbc.co.uk/1/hi/uk_politics/8118348.stm

[6] Centre for the protection of national infrastructure http://www.cpni.gov.uk/

[7] US DoD Architecture Framework Version 1.5, 2007

http://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_I.pdf

[8] Federal Enterprise Architecture http://www.whitehouse.gov/omb/e-gov/fea/

[9] UK MoD Architecture Framework Version 1.2, 2008 http://www.modaf.org.uk/m3

[10] Marco Casassa Mont et al, Identity Analytics - "User Provisioning" Case Study: Using Modeling and Simulation for Policy Decision Support, HP Labs Technical Report HPL-2009-57

[11] Open Security Architecture

http://www.opensecurityarchitecture.org/cms/

[12] IBM Rational Unified Process

http://www-01.ibm.com/software/awdtools/rup/

[13] SHERWOOD APPLIED BUSINESS SECURITY ARCHITECTURE

http://www.sabsa.org/the-sabsa-method/what-is-sabsa.aspx

[14] The Open Group Architecture Framework http://www.opengroup.org/togaf/

[15] Understanding NEC, UK MoD

http://www.mod.uk/DefenceInternet/AboutDefence/CorporatePublications/

ScienceandTechnologyPublications/NEC/UnderstandingNetworkEnabledCapability.htm

[16] ENISA, "Who-is-Who Directory on Network and Information Security (NIS)", Feb. 2009. http://www.enisa.europa.eu/act/sr/files/deliverables/who-is-who-directory-on-nis-ed.-2009/at_download/fullReport

[17] ENISA, "Report on the state of pan-European eIDM initiatives", Jan. 2009, http://www.enisa.europa.eu/act/it/eid/eidm-report

[18] ENISA, "Web 2.0 Security and Privacy", Dec. 2008, http://www.enisa.europa.eu/act/res/other-areas/web-2.0-security-and-privacy/web-2.0-security-and-privacy/at_download/fullReport

[19] NATO Network Enabled Capability http://nnec.act.nato.int/default.aspx

[20] BAE Systems Network Enabled Capability

http://www.baesystems.com/ProductsServices/ss_tes_atc_nec.html

[21] C4ISTAR (Command, Control, Communications, Computers, Intelligence, Surveillance, Target Acquisition and Reconnaissance)

http://www.baesystems.com/ProductsServices/ss_tes_atc_c4isr.html

[22] UK Defence Industrial Strategy White Paper 2005.

[23] NATO Architecture Framework Rev 3

http://www.nhqc3s.nato.int/ARCHITECTURE/_docs/NAF_v3/ANNEX1.pdf

[24] NEC at Thales Research and Technology (UK)

http://www.thalesresearch.com/Default.aspx?tabid=51

[25] The MiNT Network Demonstration Environment, Thales White paper, 2004

http://www.thalesresearch.com/Portals/0/NET041001.pdf

[26] Policy based Management in NEC, Thales White paper, 2004

http://www.thalesresearch.com/Portals/0/NET040502.pdf

[27] Secure Situation Awareness using Web Based Mashups, Thales White Paper, 2007

http://www.thalesresearch.com/Portals/0/VCS070501.pdf

[28] List of related Thales white papers

http://www.thalesresearch.com/Default.aspx?tabid=59

[29] The Informational Warfare Site http://www.iwar.org.uk/

[30] The Informational Warfare Site - Network Enabled Capability

http://www.iwar.org.uk/rma/resources/uk-mod/nec.htm

[31] Understanding NEC, UK MoD , JSP777

http://www.mod.uk/linked_files/issues/nec/nec_jsp777.pdf

[32] Smart Uses of Security Technology to achieve Information Age Government, Qinetiq White paper, March 2004

http://www.qinetiq.com/home/security/digital_security/white_paper_index.Par.22642.File.pdf

[33] Qinetiq NEC Activity

http://www.qinetiq.com/home/defence/defence_solutions/command_and_control/interoperability.html

[34] EDS Defence Services Approach to MoD Network Enabled Capability

http://www.edsdefence.com/uploads/1461%20NEC.pdf

[35] EDS Defence Battlespace Website

http://www.edsdefence.com/portfolio/battlespace/nec.html

[36] DSTL Press Release, CWID 09, May 2009

http://www.dstl.gov.uk/news_events/press/pr2009/cwid-09.pdf

[37] ATLAS Consortium Website and Brochure

http://www.atlasconsortium.info/

http://www.atlasconsortium.info/brochure.pdf

[38] What Is Windows Communication Foundation?

http://msdn.microsoft.com/en-gb/library/ms731082.aspx

[39] David Chappell, Introducing Windows Communication Foundation

http://msdn.microsoft.com/en-gb/library/dd943056.aspx

[40] Liu, Russell, et al, "Evolutionary Service-Oriented Architecture for Network Enabled Capability", Second International Workshop on Verification and Evaluation of Computer and Communication Systems (VECoS 2008)

[41] http://en.wikipedia.org/wiki/Identity_Assurance_Framework

[42] NIST Special Publication 800-63 v. 1.0.2

[43] US E-Authentication Credential Assessment Framework

[44] EAP Trust Framework

[45] Identity Assurance Framework

[46] "E-Authentication Credential Assessment Framework (CAF)" http://www.cio.gov/eauthentication/documents/CAF.pdf

[47] http://www.enisa.europa.eu/act/it/eid/idac-saml/at_download/fullReport

[48] New Zealand Guide to Authentication Standards for Online Services

[49] http://www.w3.org/XML

[50] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[51] http://uddi.xml.org/uddi-101

[52] http://www.w3.org/TR/wsdl#_introduction

[53] http://en.wikipedia.org/wiki/Enterprise_service_bus

[54] W3C XML Security Working Group http://www.w3.org/2008/xmlsec/

[55] W3C XML Signature http://www.w3.org/TR/xmldsig-core/

[56] W3C Canonical XML http://www.w3.org/TR/xml-c14n

[57] W3C Exclusive Canonical XML http://www.w3.org/TR/xml-exc-c14n

[58] W3C XML Encryption http://www.w3.org/TR/xmlenc-core/

[59] W3C Web Service Policy http://www.w3.org/TR/ws-policy/

[60] OASIS Web Services Secure Conversation http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-secureconversation-1.4-spec-os.html

[61] OASIS Web Service Trust http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.html

[62] OASIS Web Services Federation Framework http://docs.oasis-open.org/wsfed/federation/v1.2/ws-federation.html

[63] OASIS Web Services Reliable Messaging v1.2 http://docs.oasis-open.org/ws-rx/wsrm/200702

[64] OASIS Web Services Reliable Messaging Policy Assertion http://docs.oasis-open.org/ws-rx/wsrmp/200702

[65] OASIS Web Services Discovery http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html

[66] OASIS SOAP over UDP http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/os/wsdd-soapoverudp-1.1-spec-os.html

[67] Perspectives on the dependability of networks and services, Bjarne E. Helvik, Telektronikk 3.2004

[68] SOA, Dependability, and Measures and Metrics for Network Enabled Capability D.J. Russell, N. Looker, J.Xu, 2006

[69] Web Services service level management: overview of service level agreement languages, support infrastructures and tools, Tuomas Nurmela, Seminar on Service Oriented Systems, 2006

[70] Specifying and monitoring service level agreements for Web services, The WSLA Framework., Keller, A., Ludwig, H., Journal of Network System Management, 2003

[71] Building Accountability Middleware to Support Dependable SOA, LLAMA Project, Kwei-Jay Lin, Mark Panahi, Yue Zhang, Jing Zhang, Soo-Ho Chang, University of California, Irvine, 2009

[72] Fault tolerant web services,C.-L. Fang, D. Liang, F. Lin, and C.-C. Lin.. J. Syst. Archit., 2007

[73] Using WS-BPEL to Implement Software Fault Tolerance for Web Services, EUROMICRO 2006

[74] A Fault Tolerance Approach for Enterprise Applications, Vina Ermagan, Ingolf Krüger, Massimiliano Menarini, SCC 2008

[75] Sustaining Web Services High-Availability Using Communities, Zakaria Maamar, Quan Z. Sheng, and Djamal Benslimane, ARES 2008

[76] Ws-replication: a framework for highly available web services , J. Salas, F. Perez-Sorrosal, n.-M. Marta Pati and R. Jim´enez-Peris, 2006.

[77] Creating Dependable Web Services Using User-transparent Replica, Markus Hillenbrand, Joachim Götze, and Paul Müller, NWeSP 2005

[78] FTWeb: A Fault Tolerant Infrastructure for Web Services, G.T. Santos, L.C. Lung, and C. Montez, EDOC 2005

[79] High Assurance SOA-Based Systems Engineering, F.B. Bastani, R.A. Paul, and I.-L. Yen, UK-USA Workshop 2008

[80] A QoS-Aware Fault Tolerant Middleware for Dependable Service Composition, Zibin Zheng and Michael R. Lyu, IEEE 2009

[81] A Framework for Real-Time Service-Oriented Architecture, Mark Panahi, Weiran Nie, and Kwei-Jay Lin, CEC, 2009

[82] Daidalos I/II WP3 QoS

[83] Service Composition for Real-Time Assurance, Tong Gao; Moussa, H.; I-Ling Yen; Bastani, F.; Jun-Jang Jeng; COMPSAC 2008

[84] Building an Enterprise Service Bus for Real-Time SOA: A Messaging Middleware Stack, Garces-Erice, Luis, COMPSAC, 2009

[85] OASIS, Web Services Business Process Execution Language Version 2.0, 2007

[86] Architectural Support for Dependable System Evolution, Jie Xu, UK-USA Workshop, 2008

[87] NECTISE www.nectise.com