INFORMATYKA EKONOMICZNA BUSINESS INFORMATICS 3(33) • 2014
ISSN 1507-3858

Jugoslav Achkoski
Military Academy "General Mihailo Apostlski" – Skopje, Macedonia
e-mail: jugoslav.ackoski@ugd.edu.mk

Vladimir Trajkovik
Ss. Cyril and Methodius University, Skopje, Macedonia
e-mail: vladimir.trajkovik@finki.ukim.mk

DISTRIBUTED SYSTEM RELIABILITY IN INTELLIGENCE INFORMATION SYSTEM[1]

Summary: This paper presents the model of Intelligence Information System (IIS) based on a Service-Oriented Architecture. In this paper we propose the new service's model, based on the Intelligence cycle and other systems which are necessary for gathering intelligence information and data. The paper is mostly focused on the system architecture and services design as a mainstream for definition of the services. Furthermore, additional attention is dedicated on the Distributed System Reliability (DSR).

Keywords: Service-Oriented Architecture, service design, Distributed System Reliability (DSR), Intelligence Information System, system's architecture.

DOI: 10.15611/ie.2014.3.01

1. Introduction

Intelligence Information System Model gives contribution in Homeland Security and Civil Military Emerging Risks assessment through the possibility of providing information in an appropriate way, by implementing pushing and pulling mechanisms into information systems, selection of data and creation of information from raw data that can be used in creating intelligence products and dissemination reports for the authorities.

In the phase of system development, it is crucial to determine appropriate methodology which should meet different requirements. In this paper we are mostly focused on describing architecture of the system, service design and Distributed System Reliability. These three points can be used as pillars for SOA based system development.

[1] Selected parts of this article were published under nonexclusive copyright in Position papers of the Federated Conference on Computer Science and Information Systems FedCSIS 2014 (see [Achkoski, Trajkovik 2014]).

We take into consideration architecture of the system, because it will give clear picture on collaboration between system's entities, applications, programming interfaces, connection to external systems etc., in each phase of system development. Each of these components influences successful implementation and integration of the system. In the Intelligence Information System, which is based on SOA, there are applications written in different programming languages. The service design should provide interoperability between applications. It indicates that services written in different programming languages are capable to communicate. In this connotation, processing elements (Web servers, application's servers, sensors for collecting data and so on) can create distributed environment for sharing information. SOA based multi-tier approach provides legacy systems to be hooked up in a new infrastructure where new systems and legacy systems can communicate without complexity in communication protocol (SOAP messages).

In the phase of creating distributed systems it is crucial to have metrics for Distributed System Reliability. It provides appropriate distribution of the system's components, because introduced algorithm for Distributed System Reliability shows where the gaps in the systems are. Also, the designers of systems can achieve higher level of system reliability using the Distributed System Reliability metric. In the distributed system, each node can present service and each service can present system, sub-system or processing element. Consequently, each service in the architecture has certain value of reliability. These values of reliability are very important for system designers.

The paper is organized as follows. Section 2 presents related work about the research presented in the paper. Section 3 is dedicated to the architecture of the system. In Section 3 are presented different levels of the system architecture and how they are connected. Section 4 presents service design where it is mostly focused on service interface. Section 5 demonstrates the algorithm for computing Distributed System Reliability and we implement GEAR as an algorithm for the system reliability. Finally, in the Section 6, concluding remarks of the paper are presented.

2. Related work

In [Gebhart, Abeck 2011], the quality attributes of loose coupling and autonomy for services in the context of Service-Oriented Architecture are given. In order for services to be influenced by these quality attributes, an evaluation should be done during the phase of development of service design. According to M. Gebhart and S. Abeck [2011], the recent research is focused on the textual description of the desired quality attributes and the thereby resulting formalized metrics require more information than those already available, or are based on theoretical models that hamper their applicability. In this paper, we present quality indicators for unique categorization, loose coupling, discoverability and autonomy. Formalized metrics is created for each quality indicator, in order to measure service candidates and service

design in the Service oriented architecture Modeling Language (SoaML) [OMG 2009], the standardized language for modeling service-oriented architecture. To illustrate the metrics and to verify their validity, service candidates and service designs of a campus guide system as developed at the Karlsruhe Institute of Technology, are evaluated.

In [Dai et al. 2003], a study of service reliability and availability for distributed systems is presented. The study gives an application example in order to explain usefulness of the GEAR algorithm. Furthermore, in the paper is presented research about reliability of modeled centralized heterogeneous distributed system (CHDS). Also, in the paper is studied implementation of availability function of virtual machine.

3. Architecture of Intelligence Information System

General architecture of Intelligence Information System prototype is presented in Figure 1. As a result of system complexity, the solution is presented as a layer model of architecture.

On the lowest level, IIS prototype has distributed system which consists of heterogeneous databases. In this case, most important database for IIS is database which holds data for users who use it. Intelligence center has responsibility for this database.

Access to a separate database will be made with application logic of module, which is a part of internal information systems on government institution. This application should provide interfaces to the integration logic level [Achkoski et al. 2011; Achkoski, Trajkovik 2011].

Integration level is a key level for our IIS model. That level should provide services through workflow which will be connected with modules of internal information systems and their transformation into Web services. As a result of provided Web services, integration level should exposed them into appropriate Web services registers depending of security level. This level also administers security polices and polices for exchanging and adopting messages from different sources, in case of usage in comparable format. Finally, this level is taking care for governance of the services offered by IIS in a way of transactions when it is needed. In one sentence: this level is providing the functionality of the services in IIS.

The services should be available for different categories of users. For the purposes of protecting Intelligence Information System, firewall should be installed behind this level, which is followed by the level of presentational logic.

The presentation level can be implemented in a form of portal which can offer: list of Web services over approach to service registries, integration of Web services with e-mails or directly as far procedure call of the applications (RPC) in a standard format (XML), but also as an ordinary HTML text for separated union of services – users.

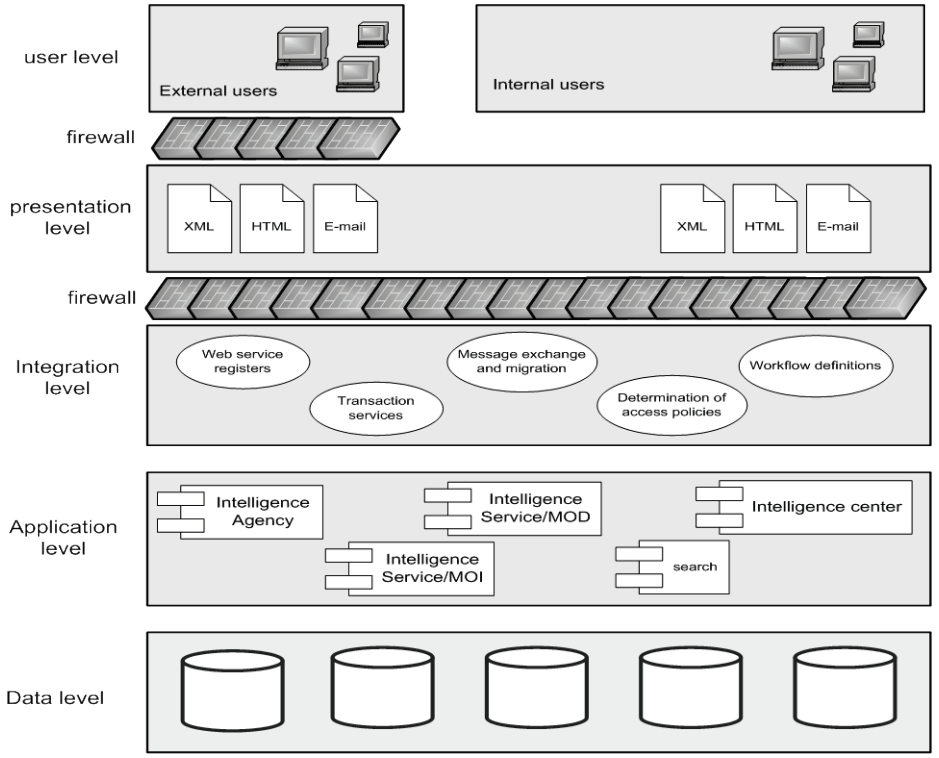

Figure 1. Prototype of Intelligence Information System Architecture

Source: own elaboration.

Exchanging information with external information systems is achievable through communication network, where IIS model is protected with another additional firewall. In this way, maximum protection from unexpected system failures is accomplished [Achkoski et al. 2011; Achkoski, Trajkovik 2011].

4. The concept of service engineering

In the process of service design it is possible to implement services in an existing platform or in a new one which will be created as a state-of-the-art solution with straightforward purpose. This distinction is important, because our model allows Web service to be implemented in both of these cases. Engineering the services in such a way allows for easy building of novel modules within information systems architecture and additionally, allows for taking a pace with a contemporary ICT technology.

We propose coupling as a way for measuring service design. In the Intelligence Information System achieved desired level of coupling allows for the integration of

subsystems and sensors as service providers with minimum number of connections between services. For example, if a new sensor is added to the system, the communication that should be established between sensor and application server or other processing elements in the system does not imply that every server should establish communication with the new sensor.

In terms of service granularity (scope of functionality exposed by the services), it is the most convenient to create coarse-grained interfaces that implement a complete business process.

The coarse-grained interface should provide access to the data from different software artifacts and processing elements in the system depending of the user requirements [Papazoglou, Heuvel 2006]. It indicates that in the IIS sensors and other hardware components, which should be connected, have to exchange information in order to provide information for the senior decision makers or other end users.

These components are based on different programming languages (C++, JAVA, C and so on) and if the service's interface is not implemented in the applications, they could not exchange their data types (see Figure 2).

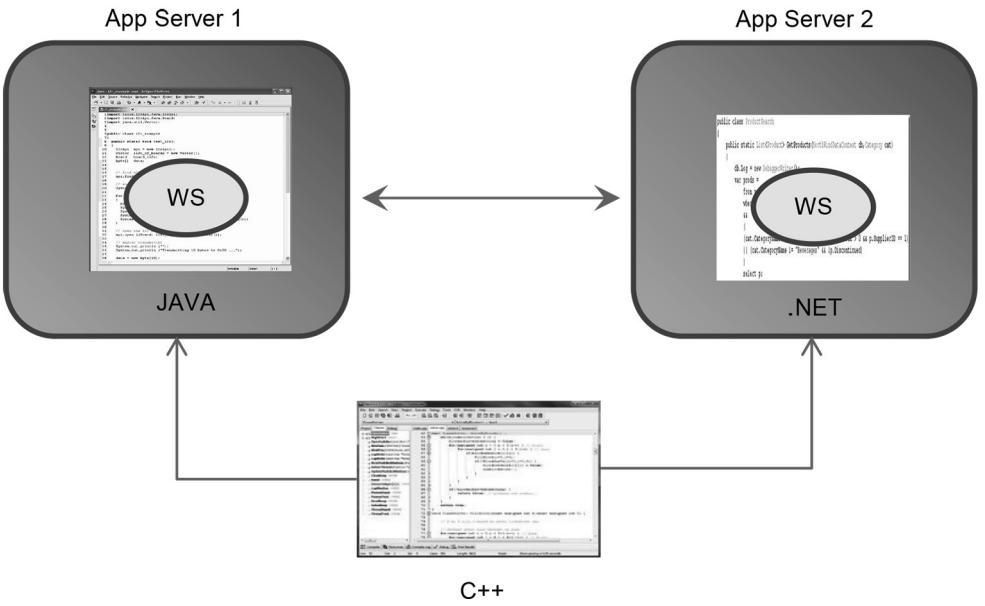

Figure 2. Exchanging data between applications written in different program languages

Source: own elaboration.

For instance, if application is written in JAVA and interface for this application is created in JAVA, then application written in C++ could not use this JAVA interface, because data types (string, integer, float and so on) in JAVA and C++ are not treated

in the same way. In our approach, the service's interface is based on XML, because applications written in different languages can exchange their data using WSDL (Web Service Definition Language). WSDL provides interoperability between applications.

Also, the standalone client application has to have functionality to call Web services. This functionality is provided by the directories called UDDI. UDDI stands for Universal Description, Discovery, Integration. This is a registry, where new businesses create a WS that can be exploited from the terminals or different machines that can actually query this directory and they can use the WS. First of all, the Web service should be published with WSDL and after that it can be exploited.

We can stress that each Web service has a public interface, WSDL (Web Services Description Language), and through simple mechanism called UDDI which is the directory for services, where machines can access to the published and registered WS with WSDL and interested parties can locate its public interface via UDDI.

Furthermore, the SOAP (Simple Object Access Protocol) which is a xml based protocol, provides interface for applications to exchange information via http:// ().

The detailed Web service process follows the scheme in Figure 3.

Discovery – Search UDDI site(s) for the proper Web service.

Description – A description of the selected Web service is returned to the client application as a Web Services Description Language (WSDL) file.

Proxy creation – A local proxy to the remote service is created. The proxy converts an object's means of method invocation into an XML message, and vice versa.

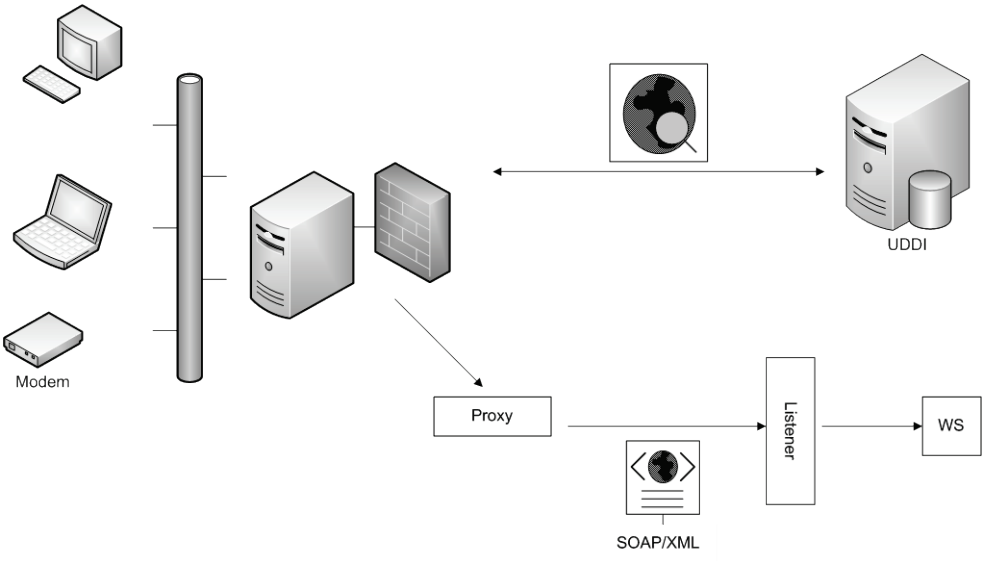

Figure 3. Web service process

Source: own elaboration.

Soap Message Creation – a Soap/XML message is created and sent to the URL specified in the WSDL file.

Listener – A Soap listener at the host site receives the call and interprets it for the Web Service.

The Web service performs its function, and returns the result back to the client, via the listener and the proxy.

The Service-oriented architecture Modeling Language (SoaML) specification defines UML profile and meta-model for designing services within service-oriented architecture. Goals of SoaML refer to support activities at the stage of modeling and designing services and invoke them in model-driven development approach (MDA). It should support SOA in business and IT perspectives [Elvesæter et al. 2011; Gebhart et al. 2010].

SoaML specification defines three different types of approaches to specifying services:

- The simple interface based approach uses an UML interface to specify a one-way service interaction [Elvesæter et al. 2011; Gebhart et al. 2010].
- The service contract based approach extends an UML collaboration to specify a binary or *n*-ary service interaction [Elvesæter et al. 2011; Gebhart et al. 2010].
- The service interface based approach extends a UML class to specify a binary or *n*-ary service interaction [Elvesæter et al. 2011; Gebhart et al. 2010].

Different SoaML approaches recommend usage of divided UML parts which mean that reading SoaML specification is not understandable. Because of the reasons

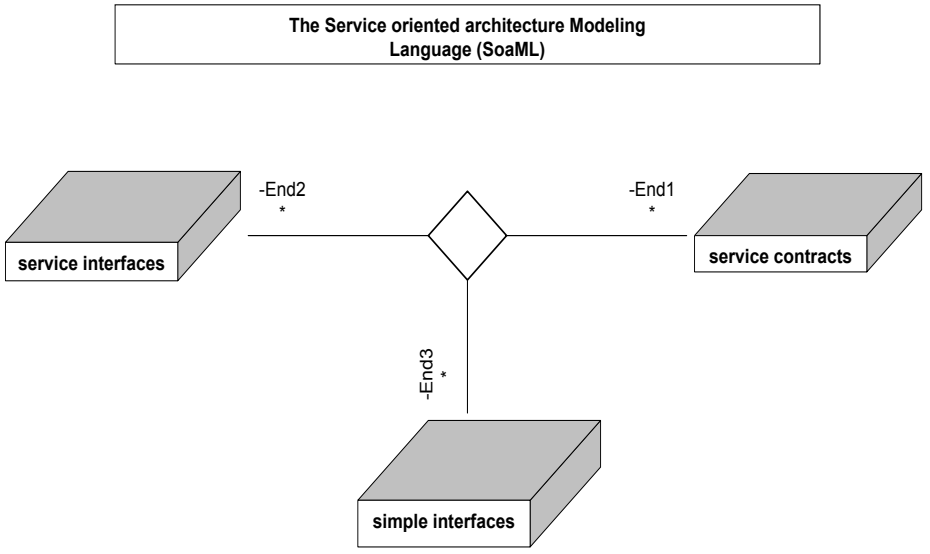

Figure 4. The Service-oriented architecture Modeling Language (SoaML)

Source: own elaboration.

previously mentioned, problems in designing information systems emerge in software engineering [Elvesæter et al. 2011].

A service contract based approach defines service specifications that define functions of service stakeholders (consumer and provider) and interface that implements these functions. Interfaces are types of ports in service-oriented architecture that requires each stakeholder to accomplish its task in the appropriate service contract [Elvesæter et al. 2011].

The service contract based approach increases the UML collaboration in the model that present structured part of services' interactions. It can be used for specifying services that include contractual obligation, i.e. an agreement between two or more parties, which is relevant for circumstances of already established interaction patterns between the participants. These interaction patterns are used for exchanging messages and specifying interfaces between participants [Elvesæter et al. 2011].

In order to demonstrate service contract based approach we are using services that make part of Intelligence Information System. Services should contribute to defining a binary service contract, a multi-party service contract and a compound service contract that contributes to explaining service contract based approach. First, we suppose that Direction service contract can be modeled as two independent

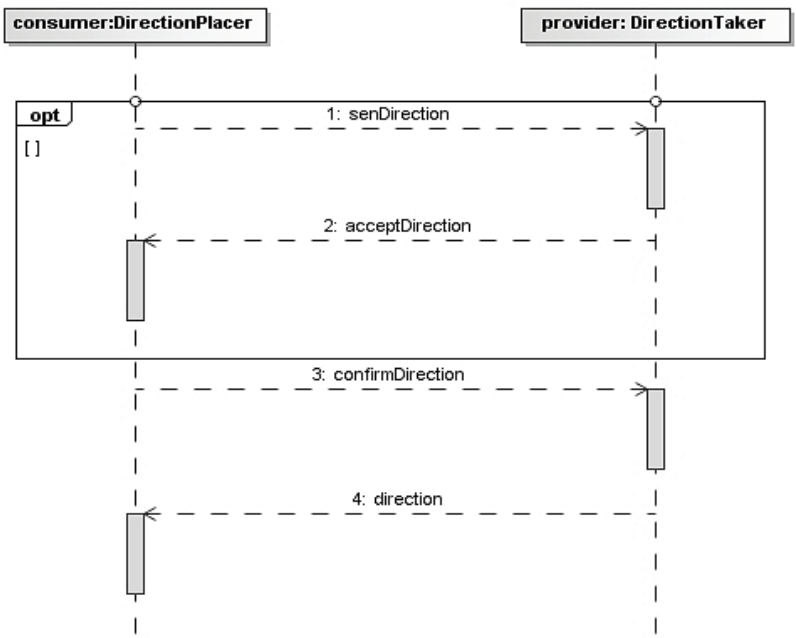

Figure 5. Specification of the Direction service choreography (behavior)

Source: own elaboration.

service contracts. One of them should specify an interaction for placing directions and another one should specify an interaction for taking directions in the process of intelligence information collection.

In this example service contract presents packing of two interfaces, providing that two interfaces are part of one service specification. Furthermore, it is recommended that a behavior on service contract is specified, i.e. a service choreography or a service protocol. Actually, there is a disagreement on whether a specification of service choreography should be used for understanding design of service interface in order to support exchanging message. SoaML is agnostic with regard to behavioral modeling and basically states that any UML behavior, e.g. interaction models, activity models or state machines, can be used [Elvesæter et al. 2011].

Figure5 shows specification of service choreography using an UML interaction. It is easy to notice that there is specified conversation between two stakeholders that exchange messages with each other. In order to achieve this conversation, it is necessary that two interfaces are implemented on both sides.

The service contract based approach is convenient for specifying interaction between two or more roles that are introduced for establishing an agreement such as, for example, a message exchange. Service contract can be also applied as a reusable specification element, which can be re-used during the design time for connecting different stakeholders. In addition, this approach supports modeling of multiparty service contracts including three or more participants, as well as modeling of compound service contracts where the existing service contract can be used for defining several granular service contracts [Elvesæter et al. 2011].

5. The Distributed System Reliability of the Intelligence Information System

The purpose of the GEAR algorithm is to compute accuracy of distributed computer system, which actually is composed of memory units, processing elements, and other hardware and software. Probability of application or service to be accurate executed in distributed system is called availability of the distributed system.

In one distributed system, the nodes can present memory units, processing elements and programs (see Figure 6). The nodes can exchange data through communication network in order to execute a program from separate nodes. Failures of the communication links or failures of the services in the distributed computer system decrease the level of system performance and availability of the system. The level of success of one program in one node in distributed computer system depends on availability (successful program execution) of all other nodes, which indicates that the node has to be accessible from all other programs required for the appropriate program execution and also, communication links should be available without failures.

To compute DSR for IIS, we select GEAR which is dedicated for computing reliability of links in computer networks. We introduce the following assumptions:

- RV maintains the information about links in the computer network [Dai et al. 2003];
- if link is operational, it has value 1 and if not, it has value 0 (faulty) in reliability expression;
- the "d" presents link when we do not know whether link is operational or faulty and it is not in reliability expression;
- LV has information about the edges that are traversed in the subnetwork [Dai et al. 2003];
- each service presents self-contained processing element, and we can assume that they are self-contained node;
- the structure of IIS is modeled as a graph and the graph does not have any loops [Dai et al. 2003; Achkoski, Trajkovik 2011].

In Figure 7 there is presented the graph topology of a distributed system, which is based on scheme from Figure 6.

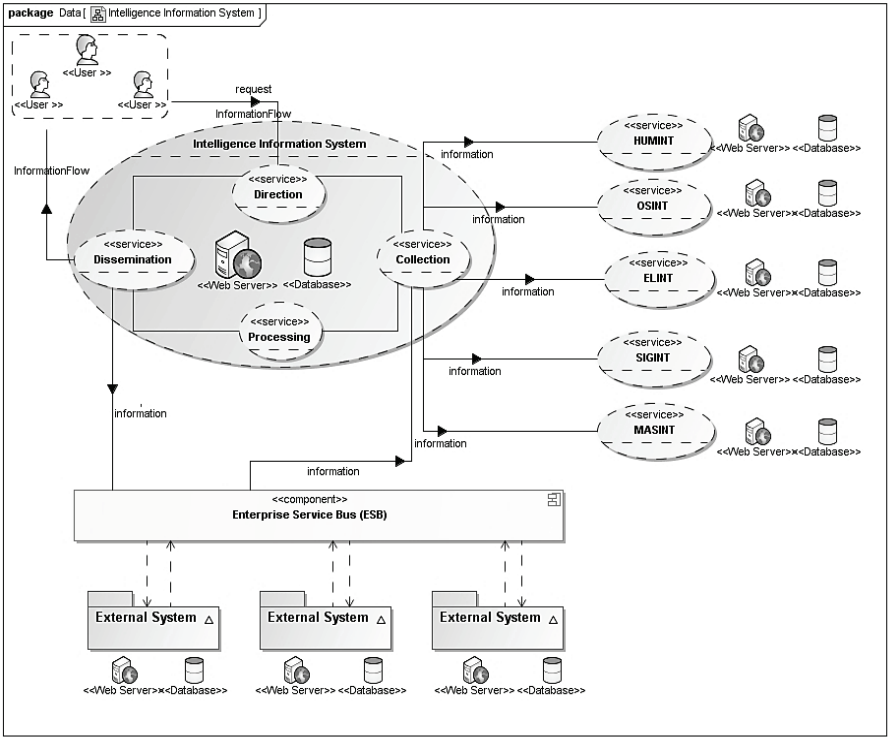

Figure 6. The dataflow in the Intelligence Information System – based on SOA

Source: own elaboration.

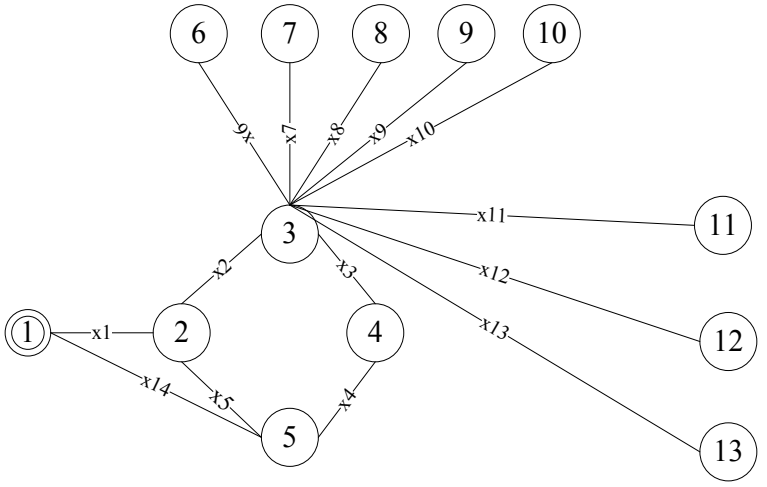

Figure 7. The graph of the Intelligence Information System – based on SOA

Source: own elaboration.

Table 1. Relations between an edge, a service name and vertices

A Node	A service name	A link	Node vectors
1	End-user	x_1	1-2
2	Direction	x_2	2-3
3	Collection	x_3	3-4
4	Analyzing	x_4	4-5
5	Dissemination	x_5	5-2
6	HUMINT	x_6	3-6
7	OSINT	x_7	3-7
8	ELINT	x_8	3-8
9	SIGINT	x_9	3-9
10	MASINT	x_{10}	3-10
11	External System	x_{11}	3-11
12	External System	x_{12}	3-12
13	External System	x_{13}	3-13
14	Dissemination	x_{14}	5-1

Source: own elaboration.

To compute the reliability of computer distributed system, the GEAR algorithm requires both vectors RV and LV to be updated in each iteration at every node. To be computed these two vectors in the GEAR are implemented simple rules without complexity.

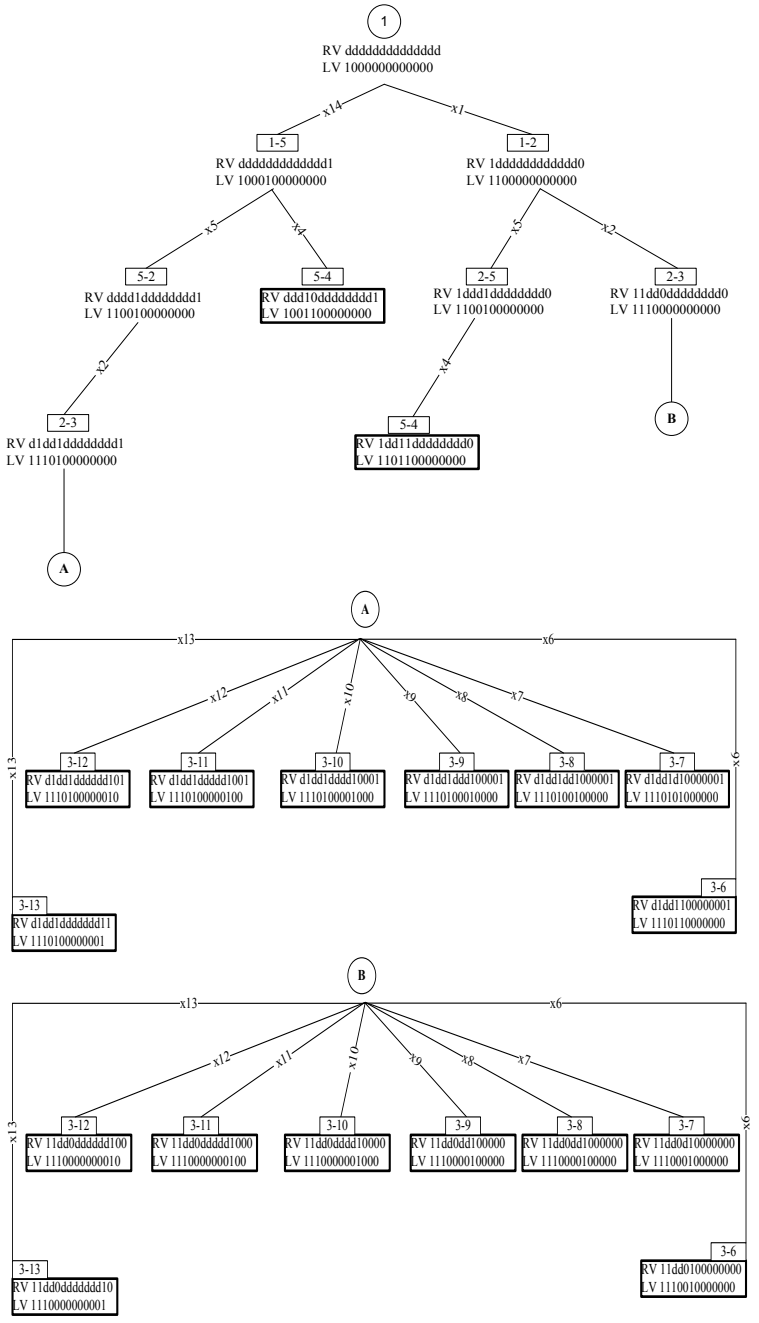

Figure 8. A complete tree for evaluating DSR in IIS

Source: own elaboration.

To update value of the Reliability Vector we have to follow the two rules [Kumar, Agrawal 1993; Dai et al. 2003]:

1. The RV updates value about the new edge which is obtained from the value of parent node and value of vertices where link is traversed from its parent edge.

2. The edges which are on the left side of the first updated edge (previous statement), will be updated with value of the edge, which is on their left side with value 0 about the vertices and value 1 about edges, which is traversed from its parent edge.

The purpose of the LV is to avoid loops in the algorithm, which means that one node is not traversed more than once [Kumar, Agrawal 1993; Dai et al. 2003]. Updating of this vector is simple, which means that every node has value from the parent node in the tree and its value represented with 1. Other nodes have value 0, and they are not connected with the parent node and the vertices of the updated node.

In order to show where in the tree GEAR algorithm stops, we set up bold rectangles (Figure 8). The tree shows that the vertices in the graph from Figure 6 have ending edges with following numbers: 4, 6, 7, 8, 9, 10, 11, 12, 13. The starting edge is number 1.

In the each ending edge (bold rectangle) we can note symbols as 1, 0 or d. In the ending where value is 1 we replaced this value with letter p and everywhere in the edge where value is 0 we replaced this value with letter q. The symbol d is not considered in computation because it is not involved in reliability expression. It allows expression to be created for DSR.

$$\begin{gathered}
DSR = p_4q_5p_{14} + p_1p_4p_5q_{14} + p_1p_2q_5p_{13}q_{14} + p_1p_2q_5p_{12}q_{13}q_{14} \\
+p_1p_2q_5p_{11}q_{12}q_{13}q_{14} + p_1p_2q_5p_{10}q_{11}q_{12}q_{13}q_{14} \\
+p_1p_2q_5p_9q_{10}q_{11}q_{12}q_{13}q_{14} + p_1p_2q_5p_8q_9q_{10}q_{11}q_{12}q_{13}q_{14} \\
+p_1p_2q_5p_7q_8q_9q_{10}q_{11}q_{12}q_{13}q_{14} + p_1p_2q_5p_6q_7q_8q_9q_{10}q_{11}q_{12}q_{13}q_{14} \\
+p_2p_5p_{13}p_{14} + p_2p_5p_{12}q_{13}p_{14} + p_2p_5p_{11}q_{12}q_{13}p_{14} \\
+p_2p_5p_{10}q_{11}q_{12}q_{13}p_{14} + p_2p_5p_9q_{10}q_{11}q_{12}q_{13}p_{14} \\
+p_2p_5p_8q_9q_{10}q_{11}q_{12}q_{13}p_{14} + p_2p_5p_7q_8q_9q_{10}q_{11}q_{12}q_{13}p_{14} \\
+p_2p_5p_6q_7q_8q_9q_{10}q_{11}q_{12}q_{13}p_{14}
\end{gathered}$$

In Figure 8, DSR is calculated with the computation of p, and q. These two coefficients present the probability of every computer network link to be available for transferring data between services with probability p = 0.9 (q = 0.1) [Dai et al. 2003]. According to expression (1) and replacement of values p and q, we can obtain the result for DSR of Intelligence Information System.

$$DSR = 0.8898.$$

In conclusion of this section, we have to stress that obtained result about DSR is not on appropriate level for the above mentioned system. Our intention is to create

the system based on Service Oriented Architecture where we can obtain result of 0.977 out of 100%. [Hurwitz et al. 2007]. In conclusion of this section, we have to stress that obtained result about DSR is not on appropriate level for the above mentioned system. Our intention is to create the system based on Service Oriented Architecture where we can obtain the result of 0.977 out of 100% [Hurwitz et al. 2007].

Furthermore, it is possible to increase the level of Distributed System Reliability, but we have to preplan the graph of computer network infrastructure with different way of connections between edges end vertices. In addition, we have to preplan the ending edges where algorithm stops.

6. Conclusion

The implementation of Service-oriented Architecture in Intelligence Information System increases the intelligence efficiency. Establishing developmental methodology and model is a basis for building of an efficient information system.

System architecture is explained in order to show general concept of the system in terms of connectivity between system components, and relationship between layers. About the system architecture we can freely conclude that level of integration logic is a basis of the Intelligence Information System. Although, in the system architecture there are diverse levels, only the level of integration logic is most significant for Service-oriented Architecture. In the level of integration logic are set up most important services for appropriate system functioning.

The service design will contribute in building Intelligence Information System based on a SOA platform because software artifacts can achieve certain level of interoperability. Therefore, diverse hardware and software can exchange their data types in order to satisfy Intelligence functions. As a conclusion about service design, we can stress that core of interoperability relies on XML because WSDL and SOAP are based on XML.

The DSR provides system's metric for reliability where many information systems rely on this metric. This metric provides reliable values for connecting nodes (processing elements, applications, I/O devices etc.) in distributed system and it can be exploited in the early stage of information system development. Furthermore, DSR could be used for gathering testing data further system development and obtained results will be taken the equations for general metric about quality of service (QoS).

References

Achkoski J., Trajkovik V., 2011, *Intelligence Information System (IIS) with SOA-based Information Systems*, [in:] *Proceedings of 33rd International Conference on Information Technology Interfaces*, IEEE, Cavtat/Dubrovnik, Croatia.

Achkoski J., Trajkovik V., 2014, *Service design and distributed system reliability in Intelligence Information System based on Service-Oriented Architecture*, [in:] Ganzha M., Maciaszek L., Paprzycki M. (eds.), *Position papers of the 2014 Federated Conference on Computer Science and Information Systems*, Annals of Computer Science and Information Systems, vol. 3, Polskie Towarzystwo Informatyczne, Warsaw, Institute of Electrical and Electronics Engineers, New York City, pp. 211–217.

Achkoski J., Trajkovik V., Davcev D., 2011, *Service-Oriented Architecture concept for intelligence information system development*, [in:] *Proceedings of 3rd international conferences on advanced service computing service computation 2011 (IARIA)*, Rome, Italy.

Dai Y.S, Xie M., Poh K.L., Liu G.Q., 2003, *A study of service reliability and availability for distributed systems*, Elsevier, Reliability Engineering & System Safety, vol. 79, no. 1, pp. 103–112, http://dx.doi.org/10.1016/S0951-8320(02)00200-4.

Elvesæter B., Berre A.-J., Sadovykh A., 2011, *Specifying services using the Service oriented architecture Modeling Language (SoaML): A baseline for Specification of Cloud-based Services*, [in:] *Proceedings of 1st International Conference on Cloud Computing and Service Science (CLOSER 2011), 7–9 May 2011*, http://closer.scitevents.org/.

Gebhart M., Abeck S., 2011, *Metrics for evaluating service designs based on SoaML*, International Journal on Advances in Software, vol. 4, no. 1/2, pp. 61–75, http://iariajournals.org/software/.

Gebhart M., Baumgartner M., Oehlert S., Blersch M., Abeck S., 2010, *Evaluation of service designs based on SoaML*, [in:] Hall J., Kaindl H., Lavazza L., Buchgeher G., Takaki O. (eds.), *Proceedings of the Fifth International Conference on Software Engineering Advances (ICSEA)*, pp. 7–13, doi: 10.1109/ICSEA.2010.8.

Hurwitz J., Bloor R., Baroudi C., Kaufman M., 2007, *Service Oriented Architecture (SOA) for Dummies*, John Wiley & Sons, Hoboken, NJ.

Kumar A., Agrawal D.P., 1993, *A generalized algorithm for evaluating distributed-program reliability*, IEEE Trans. Reliability, vol. 42, no. 3, pp. 416–426, doi: 10.1109/24.257825.

OMG, 2009, *Service oriented architecture modeling language (SoaML) – specification for the UML profile and metamodel for services (UPMS)*, Version 1.0 Beta 1, http://www.uio.no/studier/emner/matnat/ifi/INF5120/v10/undervisningsmateriale/09-12-09-SoaML.pdf.

Papazoglou M.P., van den Heuvel W.-J., 2006, Service-oriented design and development methodology, *International Journal of Web Engineering and Technology*, vol. 2, no. 4, pp. 412–442.

NIEZAWODNOŚĆ SYSTEMU ROZPROSZONEGO W SYSTEMACH INFORMACJI WYWIADU

Streszczenie: W artykule przedstawiono model Systemu Informacji Wywiadu (*Intelligence Information System*), opartego na architekturze zorientowanej na usługi (*Service-Oriented Architecture*). Zaproponowano model nowej usługi oparty na cyklu wywiadowczym i innych systemach, które są niezbędne do gromadzenia informacji i danych wywiadowczych. Skoncentrowano się głównie na architekturze systemu i projektowaniu usług jako podstawach definiowania usług. Ponadto zwrócono uwagę na niezawodność systemu rozproszonego.

Słowa kluczowe: architektura zorientowana na usługi, projektowanie usług, niezawodność systemu rozproszonego, System Informacji Wywiadu, architektura systemu.