VILNIUS UNIVERSITY

Andrius

VALATAVIČIUS

ENTERPRISE APPLICATION INTEROPERABILITY EVALUATION USING AUTONOMIC COMPUTING

SUMMARY OF DOCTORAL DISSERTATION

Natural Sciences,
Informatics N 009

VILNIUS 2019

This dissertation was prepared during 2014–2018 at Vilnius University.

Academic supervisor:
Prof. Dr. Saulius Gudas (Vilnius University, Natural Sciences, Informatics – N 009).

Academic consultant:
Prof. Dr. Audrius Lopata (Vilnius University, Natural Sciences, Informatics – N 009).

The doctoral dissertation will be defended at a public meeting of the Dissertation Defence Panel:

Chairman:
Prof. Dr. Olga Kurasova (Vilnius University, Natural Sciences, Informatics – N 009).

Members:
Prof. Habil. Dr. Rimantas Barauskas (Kaunas University of Technology, Natural Sciences, Informatics – N 009),
Prof. Habil. Dr. Antanas Čenys (Vilnius Gediminas Technical University, Technology Sciences, Informatics Engineering – T 007),
Prof. Dr. Raimundas Matulevičius (University of Tartu, Natural Sciences, Informatics – N 009),
Prof. Habil. Dr. Leonidas Sakalauskas (Vilnius University, Natural Sciences, Informatics – N 009).

The dissertation will be defended at a public meeting of the Dissertation Defence Panel at 13:00 p. m. on the 27th of September, 2019 in Auditorium 203 of the Institute of Data Science and Digital Technologies of Vilnius University.
Address: Akademijos g. 4, LT-04812 Vilnius, Lithuania.
The summary of the doctoral dissertation was distributed on the 27th of August 2019.
The text of this dissertation can be accessed at the library of Vilnius University, as well as on the website of Vilnius University:
https://www.vu.lt/lt/naujienos/ivykiu-kalendorius

VILNIAUS UNIVERSITETAS

Andrius

VALATAVIČIUS

TAIKOMŲJŲ PROGRAMŲ SĄVEIKUMO VERTINIMAS TAIKANT AUTONOMINIO SKAIČIAVIMO TECHNOLOGIJAS

DAKTARO DISERTACIJOS SANTRAUKA

Gamtos mokslai,
informatika N 009

VILNIUS 2019

Disertacija rengta 2014-2018 metais Vilniaus universitete.

Mokslinis vadovas:
prof. dr. Saulius Gudas (Vilniaus universitetas, gamtos mokslai, informatika – N 009).

Mokslinis konsultantas:
prof. dr. Audrius Lopata (Vilniaus universitetas, gamtos mokslai, informatika – N 009).

Gynimo taryba:

Pirmininkė:
prof. dr. Olga Kurasova (Vilniaus universitetas, gamtos mokslai, informatika – N009).

Nariai:
prof. habil. dr. Rimantas Barauskas (Kauno technologijos universitetas, gamtos mokslai, informatika – N009),
prof. habil. dr. Antanas Čenys (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – T007),
prof. dr. Raimundas Matulevičius (Tartu universitetas, gamtos mokslai, informatika – N009),
prof. habil. dr. Leonidas Sakalauskas (Vilnius universitetas, gamtos mokslai, informatika – N009).

Disertacija ginama viešame Gynimo tarybos posėdyje 2019 m. rugsėjo mėn. 27 d. 13 val. Vilniaus universiteto Duomenų mokslo ir skaitmentinių technologijų instituto 203 auditorijoje.
Adresas: Akademijos g. 4, LT-08412 Vilnius, Lietuva.
Disertacijos santrauka išnagrinėta 2019 m. rugpjūčio mėn. 27 d.
Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje ir Vilniaus universiteto interneto svetainėje adresu: *https://www.vu.lt/naujienos/ivykiu-kalendorius*

SUMMARY

1. INTRODUCTION

1.1. Research area

Application interoperability evaluation is required for gaining knowledge on whether different software applications could exchange data between one another. Application services describe the data structure of these applications. Analysis of such application service descriptions allows us to infer whether different applications have some common ground from a data perspective. Syntactical and semantical description documents of application web service were analyzed in this research. Similarity information between operations, objects, field names and field types were retrieved from web services and analyzed using the edit distance, bag of words and latent semantic analysis methods. The autonomic computing in the theoretical part of the research presents a broader picture of possibilities of the implemented research. The results of the experiment only cover three parts of the autonomic component: monitoring, analysis, and knowledge. The autonomic computing component was introduced for analysis of the possibility of an automating interoperability process within the dynamic business environment.

1.2. Relevance of the problem

Application interoperability becomes the essential part for dynamic business, growing IOT usage and ever-growing complexity and variety of enterprise applications. Enterprise applications are now used in almost all medium to large sized companies, and interoperability projects are becoming relevant because of the need to optimize business process, reduce redundant work, and increase the efficiency of data maintenance along with different applications within an enterprise. The challenge is that the knowledge requirements for integrating different systems are great and there is a high risk of failure of integration and interoperability projects. To measure the potential of applications to be interoperable first of all we need to evaluate their capability of interoperability.

In informatics, the interoperability subject is quite old and stems from the requirement that devices, satellites or other military or civil equipment should be able to exchange data. For example, it is important that NASA (Di & Kobler, 2000) has satellites that are able to communicate with the ground stations and exchange important telemetry data. Application integration and interoperability projects have a tendency to fail at almost 70% (Trotta, 2003; van der Bosch, et al., 2010), mainly due to lack of knowledge of the application, growing complexity and dynamic nature of business. The interoperability process is analyzed and classified into different levels: syntactic, semantic, and cross domain (Chen, et al., 2008). Each level concerns different issues of interoperability solution. According to (Rezaei, et al., 2014), there are different granularity issues for interoperability; scientists had reviewed the complexity of the subject and techniques by 2014.

1.3. The aim and tasks of the research

The goal of this research is to create a method for enterprise application interoperability evaluation based on causal relationships extracted by comparing architectures.

The object of this research is an enterprise whose business process is dynamic (changing) and where the use of applications from more than one provider might face interoperability issues such as data redundancy and duplication of business processes.

To realize the aim of research the main tasks were established:

1. To analyze the problems of the enterprise application integration and interoperability solutions, applied methods and their principles.
2. To analyze the methods of the enterprise application interoperability, their advantages and flaws, and underline the principles of the proposed method.
3. To create enterprise application capability of an interoperability evaluation method using business process architecture (CIM – computation independent models) and enterprise application architecture (PIM – platform independent models).
4. To perform an experiment in order to prove that enterprise applications' interoperability can be evaluated using the

proposed method for detection changes in dynamic business process and to indicate the changes of enterprise application affecting interoperability.

1.4. Scientific novelty

1. Established theory of possibilities to computationally evaluate enterprise applications interoperability by using multiple data source domains, such as business process models, autonomic computing, deep knowledge extraction from application web service architecture descriptions.
2. Proposed the text processing method for enterprise application interoperability capability evaluation; capability evaluation depends on text processing methods such as edit-distance, latent semantic analysis, bag of words.
3. Applied edit-distance methods: Levenshtein, Jaro-Winkler, Jaccard, and Longest Common Subsequence; the achieved results show each application capability to interoperate with another application.
4. Applied latent semantic analysis for better semantic extraction capabilities from application web service architecture to better evaluate the capability of applications to be interoperable.

1.5. Statements to be defended

1. Enterprise architecture (EA) frameworks and model-driven architecture (MDA) can be applied when solving enterprise application interoperability issues, by visualizing and identifying relationships between application components and business process causal relationships.
2. Proposed enterprise applications interoperability capability evaluation solution is sufficient to evaluate similarities between applications at syntactic and semantic levels.
3. It is possible to use the CIM and PIM models to evaluate applications interoperability by extracting causal dependencies between business processes and their counterparts that are transformed to match application processes.

4. Enterprise application interoperability evaluation solution based on autonomic computing technologies enables detection changes in dynamic business processes and shows the changes affecting enterprise application interoperability.

1.6. Approbation of the research

The results of the research have been published in two peer-reviewed journals, in seven peer-reviewed conference proceedings and were presented and discussed in four national and international conferences. Intermediary results and discussions were presented at two national workshops.

1.7. Outline of the dissertation

The dissertation consists of seven chapters and a list of references. The chapters of the dissertation are as follows: Introduction; Review of enterprise application interoperability solutions; Measures of enterprise application interoperability; Application interoperability evaluation experiment description; Results of application interoperability evaluation experiment; Conclusions and recommendations. This work contains 83 pages that include 28 figures and 8 tables; the list of references consists of 55 sources.

2. REVIEW OF ENTERPISE APPLICATION INTEROPERABILITY SOLUTIONS

In this chapter enterprise application interoperability and integration solutions are reviewed. Methods that solve interoperability and integration problems are described. A list of main interoperability problems is compiled. These methods are used to make applications integrated or interoperable within a business domain, but they pose an issue of high maintenance and knowledge requirements that are sometimes so difficult that most integration projects fail (Trotta, 2003), thus new solutions should be proposed.

It is known that integration and interoperability of applications differ by goal business: to create a single holistic system to cover all processes, or to effectively use multiple applications that would efficiently exchange data and would not be limited to a single application provider. In other words integration encompasses the entire domain, while interoperability focuses on parts of the same domain that should effectively exchange data and functionality (Chen, et al., 2008).

In a dynamic organization, there could be multiple obstacles that do not allow legacy and new applications to interoperate automatically. Mainly these obstacles are (Fig. 1):

- Business processes change when new applications are introduced – this causes dependent process failures, data errors, time delays and has overall demanding requirements for organization adaptability.
- Applications are dynamic; their schema might be changed over time – this causes failures in schema matching, interoperability and integration solution failures, business process failures and time delays.
- Multiple applications are used in a single domain – this causes data ambiguity and duplication between those applications, and new processes appear to solve these issues, causing higher human resource requirements.
- There are no common methods to describe collaboration among multiple different applications – this causes ambiguity, different application architecture strategies, new integration protocols development or requirements for heightened maintenance.

- Application changes usually impact business process. Therefore, the previous business process models become invalid and cannot be used for knowledge extraction – this is caused by one-time modeling, and therefore after some time the model could not represent the current status of an enterprise.
- To ensure interoperability, the integration expert needs to perform the following tasks:
  - Perform schema alignment (Hophe & Woolf, 2004), (McCann, et al., 2005) (Peukert, et al., 2012), (Rahm & Bernstein, 2001), (Silverston, et al., 1997), (Silverston, 2011);
  - Ensure record linkage and data fusion (Dzemydienė & Naujikienė, 2009), (Kasunic, 2001)
  - Ensure orchestration – the timing of each data migration;
  - The choreography of application services and data objects – sequence and order in which applications could share data.
- Lack of skills and knowledge – this causes integration and interoperability project delays and failures.

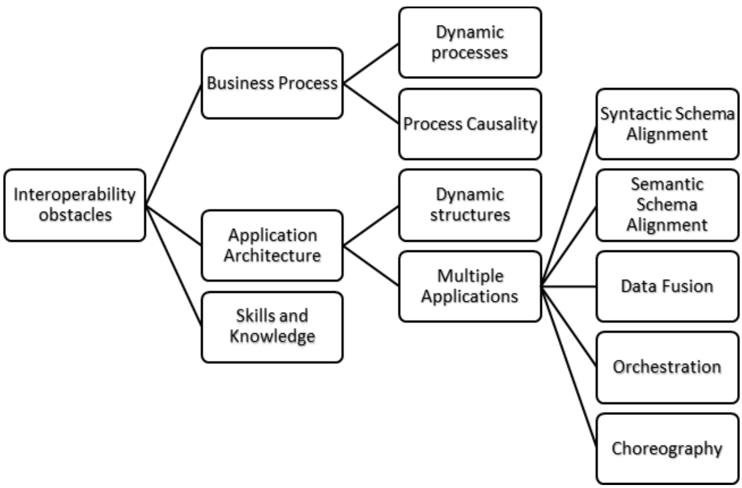

Fig. 1 Tree of interoperability obstacles.

Lack of necessary skills is a barrier to implementing interoperability solutions. Lack of the necessary knowledge of the applications used is also an obstacle to the implementation of interoperability solutions. The full tree of interoperability obstacles is represented in the Tree of interoperability obstacles figure (Fig. 1). In earlier documents of EIF (IDABC, 2008) interoperability layers were called barriers. Data from one system cannot be interoperable with similar data in another system without crossing these barriers. The five layers of interoperability are the following:

- Governance layer – decisions on interoperability structures, roles, responsibilities policies, and agreements.
- Organizational layer – these barriers relate to the structure of an organization and how this organization is dealing with constant and rapid changes. Usually, the structure of every organization and especially its processes must be discovered and evaluated. Some integration solutions can help improve business processes and therefore overcome the organizational barriers (Valatavičius & Gudas, 2015).
- Legal layer – to ensure that the data will not be abused or leaked to the public during the interoperability operations. This layer also might include, for example, a new general data protection regulation (GDPR) that allows people to get all related data from business applications.
- Semantic layer – semantic or conceptual layers cover semantic differences of information, for example, the use of different software systems leads to semantic differences.
- Technical layer is a layer in which interface specifications, communication medium, interconnection services, data integration services, and other aspects are analyzed.

The interoperability domain describes the object of the interoperability solution. As there could be multiple layers of interoperability, a different aggregation and granularity of data are taken into perspective. Interoperability areas investigated by other researchers are as follows (Chen, et al., 2008): data, services, processes, and business. The interoperability of data covers different issues of the complex data integration from diverse sources with different schemas. The interoperability of services covers different issues of the heterogeneous data covered by the shell of web services of applications that are designed and implemented independently. At this level of interoperability, it might be easier to deal with different schemas and solve semantic issues. The interoperability of processes solves the problem of process sharing or optimizing a value chain for a company. The processes are optimized by developing good interoperability of services/data that are used in these processes. Recent research showed that it might be possible to get internal models from the business process and apply it as knowledge in integration solutions (Valatavičius & Gudas, 2015). The interoperability of business covers B2B integration problems and focuses on the issues of data sharing between businesses, but all previous interoperability options must be assured to have a successful business.

3. MEASURES OF ENTERPRISE APPLICATION INTEROPERABILITY

Various application interoperability methods are applied to create and maintain the interoperability of enterprise applications. The research varies among layers (e.g., organizational, legal, semantic and technical) and levels (system specific, documented data, aligned static data, aligned dynamic data, harmonized data) of the conceptual interoperability model (Tolk & Muguira, 2003). Most researchers of the integration subject use advanced methods such as agent technologies (Cintuglu, et al., 2016; Overeinder & Verkaik, 2008) that usually cover self-describing services, which cannot be applied in the RESTful protocol in applications. Moreover, as the RESTful protocol becomes an increasingly popular API protocol in business applications, this provides a difficulty to create automated bindings between different systems. Even with a good protocol description, usually the lack of semantics could also be a blocking point for successful interoperability (Li, et al., 2005; Shvaiko & Euzenat, 2011). However, sophisticated methods of the process integration already exist, but they are not applied in the application area (El-Halwagi, 2007). In a dynamic environment, business processes often need optimizing; one of the examples being business process integration (El-Halwagi, 2007; Pavlin, et al., 2009).

Some researchers underline the guidelines of measurements and give propositions of what methods should be used, but they are not presented in such a way that could be easily replicated. One of the favorite inspirers of this research Kasunic (Kasunic, 2001) proposed to evaluate systems interoperability using three views: Technical, Operational, and Systems. A similar approach to the business and information systems alignment measurement is introduced in (Morkevičius, 2013).

Table 1. Selected systems interoperability capability measure by the LISI method.

	a) Technical view, Technical interoperability scorecard.	b) Systems view, Systems interoperability scorecard			
Source	Compliance to standards	S1	S2	S3	S4
S1 ExactOnline	Y		Y	Y	G
S2 PrestaShop	Y	Y		G	Y
S3 SuiteCRM	Y	Y	G		Y
S4 NMBRS	G	G	Y	Y	

The technical view table indicates that it needs more effort than anticipated to extract metadata (Kasunic, 2001). The colors represent the usage of standards in Table 1 above inadequate (R), marginal (Y), or adequate (G). Conclusions: such an evaluation method could be biased by one's understanding of whether the system is standardized and how easily it could integrate providing interoperability.

The enterprise application (EA) interoperability measurement (between services) is the basis for improving interoperability methods. Some interoperability evaluation methods are known: Scorecard – DoD in (Kasunic, 2001), I – Score in (Ford, et al., 2008), and Comparison by functionality in (Dzemydienė & Naujikienė, 2009).

These EA interoperability evaluation methods are not enough because the assessments are obtained through questionnaires and expert judgment. We strive to develop a method that evaluates the characteristics of the systems being integrated without using personal opinions or tests/questionnaires/experiences. We aim to use only characteristics of software: metadata and systems network service architectures. It is more reasonable to use structured (internal) models of systems than to fill out questionnaires. We are looking for a deterministic method that can evaluate or measure the capability of interoperability.

The principles of the second order cybernetics provide the methodological basis for the internal viewpoint and aim to disclose internal causal relationships of the domain. In our case, we need to explore the causal relationships between application software and there is no access to use the questionnaires as stated by (Kasunic, 2001).

1.1. Interoperability evaluation using MDA and EA approach

Our study is based on a few assumptions. First, internal modeling with the MDA approach help determine the influence of domain causality to the interoperability of applications (Fig. 2). Second, it is possible to create an architecture of interoperable enterprise applications using only the enterprise architecture model and data for each service for enterprise software. Another assumption is as follows: interoperability should be evaluated by comparing web service operation names using edit distance calculations. The measurement of EAS interoperability capability serves as a basis for improving interoperability methods. When interoperability is required between these applications, how should one know whether these systems can have interoperability at all? The capability of interoperability of applications can be evaluated with the help of their architectural design by comparing web service operation names using edit distance calculations.

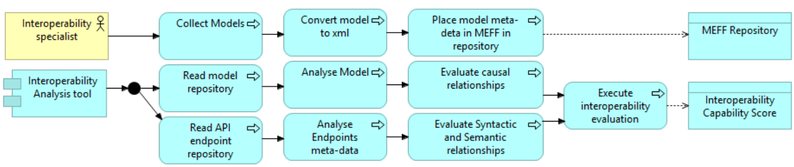

Fig. 2. Analysis of models from the MDA cycle to produce interoperability capability score.

Levenshtein calculates edit distance by a minimum number of single character edits required to change the first word into the other. The Levenshtein algorithm was the first known method developed to compare string distances in 1965 (Левенштейн, 1965): for a given two strings b and a with a total character count of m and n, and for each character pair of two strings count the minimum amount of changes required to make them similar if they are not equal. The Jaro-Winkler algorithm uses a formula out of 4 values that calculate similarity. The Longest common subsequence edit distance algorithm as the name suggests calculates edit distance by removing characters, and counting how many characters are removed to leave the longest common subsequence. The Jaccard edit distance algorithm calculates how many similar attributes there are in the two compared sets for an n-gram. For a given character sequence of each string, a character matrix is formed where characters for each set represent the total number of characters which have the same value (matched).

String distance algorithms only provide syntactic similarity evaluation capabilities. For semantic evaluation capabilities, we have developed an ontology library describing data structure with semantic meaning. The steps to calculate interoperability capability (potentiality) are the following: 1) locate web-service reference documentation; 2) extract and parse metadata of web service reference files; 3) categorize the parsed metadata into operations, methods, objects, field names, and field types; 4) select operations and create metadata for each operation: a) get the name of the source; b) get service name; c) extract methods GET, POST, PUT, DELETE, PATCH, HEAD); d) extract operation to the related method; e) strip redundant information from operation (repeating meaningless keywords; 5) save operation metadata to Microsoft SQL Server database; 6) using master data services and the prepared SQL procedure scan through operations in the database table and compare it with other operations from different source; 7) save each comparison for a different method in a new table; 8) visualize and explore the results.

For the following systems (OpenCart, PrestaShop, LemonStand, NMBRS_ReportService, NMBRS_DebtorService, Zen Cart, NMBRS_CompanyService, NMBRS_Employees, SuiteCRM, KonaKart_StoreFront, KonaKart_Administration, MIVA, ExactOnline) used in the experiment, we describe web service interface protocol and complexity to extract data automatically. According to the documentation SOAP and REST, development should follow design recommendations, but there are already many systems developed without SOA approach. Once a system implements web services, it is required to have an API which is not always created using common recommendations. Therefore, it is harder to automate data extraction. Additional steps are needed to get to the objects of web services as it is not enough to get the initial structure described in web service for metadata analysis. During the experiment, additional steps were taken invoking web service for returning the list of objects related to the operations described in SOAP WSDL files. REST web service metadata description is not standardized, and it is more challenging to extract metadata. A lack of a common pattern following the description of objects exists; therefore, additional procedures to extract and parse metadata from API are needed. The web service metadata for each system data is extracted to the database using a custom written C# algorithm and manual data entry from web service reference documentation. Data storage was setup using the Microsoft SQL Server database. From the database, data was analyzed, cleaned, and formed in such a way that it is usable with edit distance measurement algorithms. Edit distance algorithms were executed using Microsoft SQL Server Master Data Services to produce enterprise software system compatibility for interoperability result. Further results and data are described in Section 6.

1.2. Interoperability evaluation and autonomic computing

The autonomic computing technology was presented by IBM researcher Jeff Kephart (Kephart & Chess, 2003). The purpose of the technology is to raise the automation level of computing solutions. With the intention of applying the autonomic computing technology to enterprise application integration and interoperability solution it was discovered that there are big similarities between the autonomic computing and elementary management cycle from business process modeling (Gudas, 2012). The IBM autonomic computing element consists of these components:

- Touchpoints – in this research domain it is URL addresses to application API reference source.
- Knowledge – in this research domain it is application web service description documents, business process diagrams and ontology models representing the domain.
- Autonomic manager – in this research domain it is the solution for interoperability evaluation.

- Managed /resources – in this research domain it is applications that should be interoperable.

Autonomic Manager consists of five main components:

- Monitor action – which is covered in the experiment by scanning data sources in a scheduled fashion.
- Analyze action – which is covered in the experiment by determining interoperability score.
- Plan action – is not covered in this research.
- Execute action – is not covered in this research.
- Knowledge storage – which is covered in the experiment by storing intermediary results from edit-distance calculations, latent semantic analysis etc.

Autonomic computing solution is usually depicted similarly as applied IBM autonomic computing component architecture (Fig. 3). Monitor (M) reads data sources and analyzes their structure; then Analyze (A) step evaluates interoperability; Plan (P) step reads evaluation of object interoperability value and determines actions how to exchange data; Execute (E) step would initiate another autonomic component capable of starting data transfer between two or more applications, which in turn affects the application by migrating data.

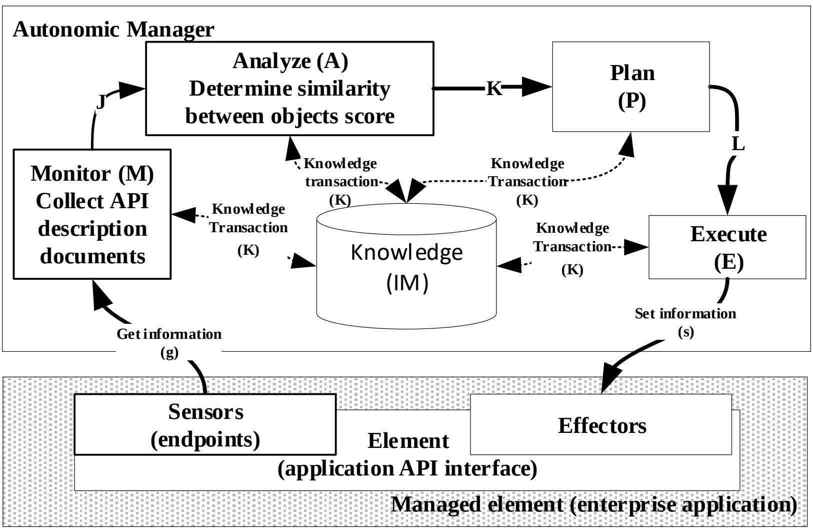

Fig. 3. IBM autonomic computing component architecture

The idea behind this solution is only valid under certain conditions:

- Application is developed with a service-oriented architecture in mind.
- Application has API that is properly described regarding standards and agreements (such as SOAP, REST protocols).
- User can provide details about the endpoint to the interoperability solution.

The items described in autonomic computing component architecture (Fig. 3) are only partially described in the dissertation and covers part of it since it was out of the scope of this research.

4. APPLICATION INTEROPERABILITY EVALUATION EXPERIMENT DESCRIPTION

This research is limited to enterprise applications developed using service-oriented architecture and mostly focus on software that uses web services and SOAP and RESTful protocol for data transfer whose metadata is usually described using standardized documents. Web service operations compared to multiple software system applications for the enterprise show the difference in similarity scoring. Randomly picked applications are presented in the table below (Table 2). Each application has some different roles and aspects of an enterprise. Although this research is limited to a few applications, the intention is to expand the research to involve more applications. The core set of applications are On-site e-commerce applications and some on-site accounting applications.

Table 2. Randomly picked software applications for analysis

Software Application	API protocol	Objects	Description
OpenCart	REST	24	On-site e-commerce application
PrestaShop	REST	49	On-site e-commerce application
LemonStand	REST	76	On-site e-commerce application
NMBRS_ReportService	SOAP	80	On-site accounting application
NMBRS_DebtorService	SOAP	106	On-site accounting application
Zen Cart	REST	208	On-site e-commerce application
NMBRS_CompanyService	SOAP	444	On-site accounting application
NMBRS_Employees	SOAP	1107	On-site accounting application
SuiteCRM	SOAP	1426	On-site CRM application
KonaKart_StoreFront	SOAP	1644	On-site e-commerce application
KonaKart_Administration	SOAP	2425	On-site e-commerce application
MIVA	REST	4322	Cloud e-commerce application
ExactOnline	REST	6043	Cloud accounting application

For these applications and their services (Table 2), API reference data is collected and parsed to evaluate interoperability. Microsoft SQL Server, PostgreSQL, R, Microsoft Visual Studio, and Tableau were used to acquire data from web services. We used Microsoft SQL Server to collect initial data from C# script written to extract and parse API reference descriptions. C# reference parser was good for a limited amount of applications, but more time was needed to enable it to work with a more extensive data set. C# script loaded metadata from API, was parsed and stored in Microsoft SQL server. Later for edit distance analysis, R script was used to determine similarities between operations, objects, and fields of sets between multiple applications. Data was stored into the PostgreSQL server. The data was finally analyzed and represented using Tableau software. The activity diagram below depicts a proposed solution of interoperability capability analysis tool (Fig. 4).

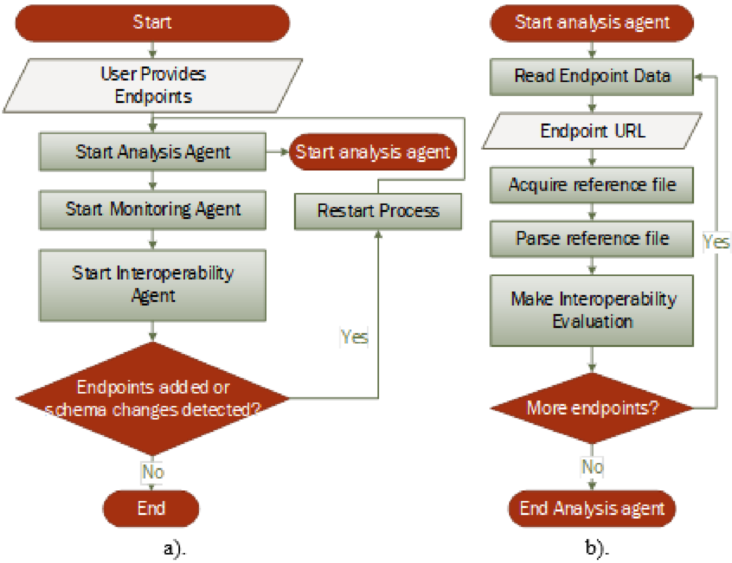

Fig. 4. Activity diagram of the proposed solution of interoperability capability analysis and interoperability tool.

The figure above (Fig. 4 b) depicts a simple process of analysis agent. This agent takes part in the job done manually by a data integration specialist. It reads the endpoint data from the endpoint URL, acquires the reference file, and then parses it and runs evaluation scripts; then repeats the entire process for more endpoints. In the holistic view of software interoperability, there should be three steps: Analysis, Monitoring and Action (interoperability), hence, the three blocks in the activity diagram (Fig. 4). The interrelation between the activity diagrams in a) and b) in the figure file is that the subactivities of the analysis agent might be running independently from any other agent activity, such as monitoring or interoperability.

5. RESULTS OF APPLICATION INTEROPERABILITY EVALUATION EXPERIMENT

For each enterprise application, it is possible to gather metadata of web service and API descriptions. Some metadata are automatically extracted from these services (therefore can be automated), other EA require more efforts to do the extraction, but with careful rethinking the metadata extraction can also be automated. Section 5 describes the interoperability capability (potentiality) evaluation experiment of 9 different enterprise software applications (see Section 5). Some of the applications are repeated in the list (Table 2) because web services have several descriptions of different packages with different endpoints. Using the metadata of web services we counted for each system how many operations can be carried out using its web services (Fig. 5).

The largest analyzed enterprise application is MIVA – a cloud computing based e-commerce application. Automated parsing determined 3,908 data related operations for this specific application. For ExactOnline and NMBRS (employees related web service) counted 293 and 265 operations respectively. KonaKart, ZenCart, SuiteCRM contained a smaller number of web service operations – below 150.

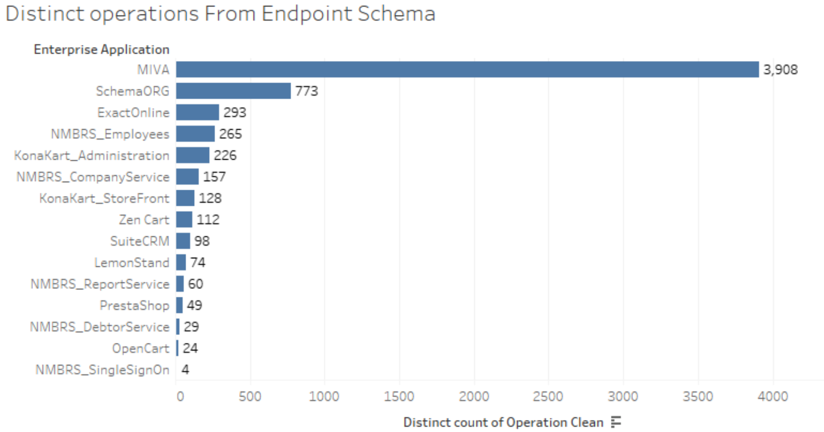

Fig. 5. Number of distinct operations in EA packages.

The number of distinct operations in the EA packages list included an additional collection of metadata from Schema.org and added background knowledge and semantics for other applications (Fig. 5).

Considering only the number of operations that can be carried out by EA packages, some conclusions can be drawn:

- MIVA the most extensive software package from a test set;
- MIVA contains more modules and data management points than other systems;
- Other systems are smaller, or their web services are limited or split (e.g., NMBRS).

There is a total of 5,323 distinct operations used in the experiment. On average, EA has 116 operations per system provided by their web service (excluding SchemaOrg and MIVA). The results of the experiment are the analysis of similarity of each operation name in each enterprise application. If the edit distance for each operation name is high enough, this indicates that most operations are similar in that pair of EAS packages. The results in Figure 5 summarize the outcome of the edit distance calculations for e-commerce packages. The heatmap of possible interoperability (Fig. 6) shows the edit distance score of operations. In the Prestashop to KonaKart_StoreFront interoperability comparison the red spots indicate < 50 % operation similarity as opposed to other operations (green); the white area indicates around 50% similarity. The red spots also indicate a higher probability of operations being similar. For example, PrestaShop operation “categories” matches KonaKart_StoreFront operation “category” by 75% using an ensemble of edit distance calculation.

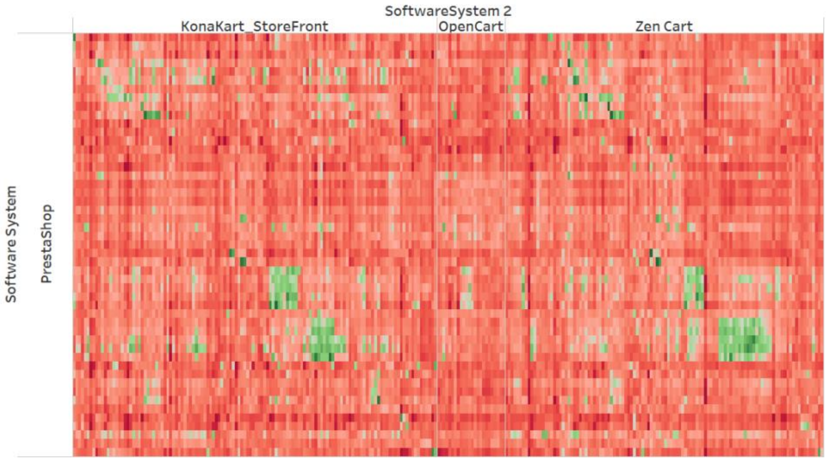

Fig. 6. Operation interoperability scoring – a heatmap using the average ensemble score of edit distance algorithms; the green spots indicate above 50% similarities.

In the operation interoperability scoring figure (Fig. 6) the similarity of operations of e-commerce products presented is apparent. In this example, syntactic overlap can compare and evaluate syntactic overlap of operations between software applications. Results from multiple edit distance methods (Levenshtein, Jaccard, Jaro-Winkler, Longest Common Subsequence) are presented further in the text. An average score of all selected methods was not in the scope of this research to evaluate edit distance methods, but rather provide an overview of the capability of evaluation.

1.3. Interoperability evaluation using ensemble method

The evaluation of the results is presented using the ensemble method. The ensemble method is the average of all similarity scores from the edit distance algorithms. After looking at the results from the operation level, we see that operations of web services are similar to each application: accounts; absences, addresses (Fig. 7). The results of the operations interoperability scoring leads to the following conclusions: In ExactOnline (E) and NMBRS (N) there exist operations that are similar: E Addresses – N Address (85%); E BankAccounts – N BankAccount (91%); E Cost centers – N CostCenter (90%); E Cost units – N CostUnit (88%); E Departments – N Department (90%); E Employees – N Employee (88%); E Schedules – N Schedule (88 %).

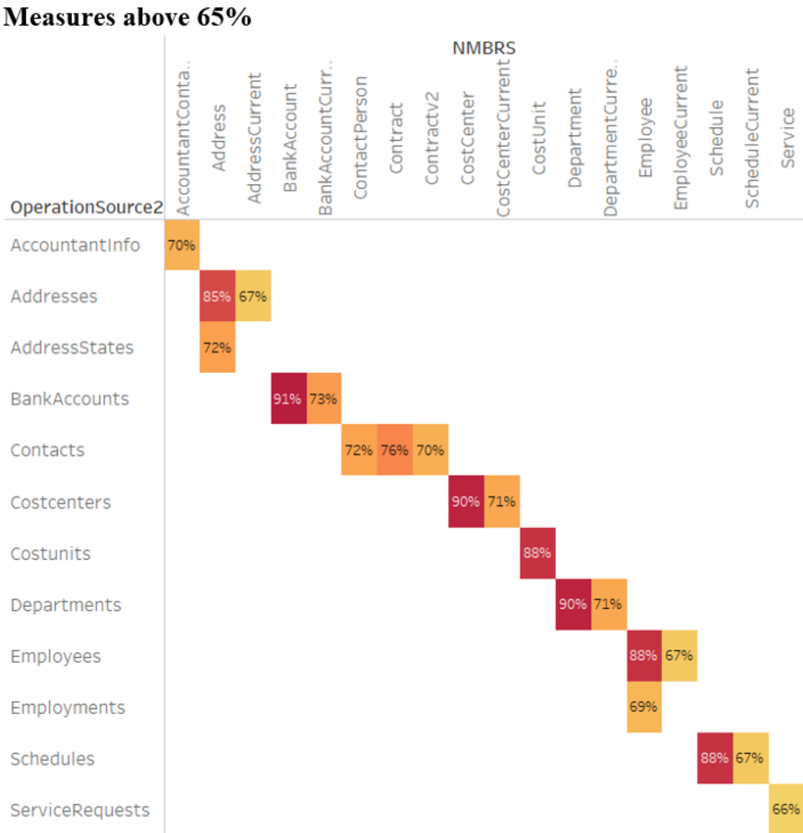

Fig. 7. Similarity results greater than or equal to 65 % (Exact Online, NMBRS).

In Exact Online (E) and NMBRS (N) there exist operations that are confused: E Contacts – N Contract (76%); E Contacts – N ContractPerson (72%) – they share some similar data, but they need to be evaluated from data structure perspective for this operation; E Contacts – N ContractV2 (70%);

Exact Online with NMBRS has 20 operations with a result higher than 65%. We can analyze and determine thresholds by semantic meaning trying to avoid mismatching. As can be seen, Exact Online 285 NMBRS 130 operations have only 20 operations possible with interoperability score > 65%. Further, Exact Online (E) and PrestaShop (P) were compared and similarity results were above or equal to 70 %. In the research results there are cases with full similarity (100%) between a few objects: Addresses; Contacts; Currencies; Employees; Warehouses. However, the algorithms are not precise, so some confusion can be found, for example, at (74%): E Projects – P products (74%).

Exact online with PrestaShop has 18 operations with a result higher than 70 %. As can be seen, Exact Online 285 PrestaShop 72 operations have only 18 operations possible interoperability with score > 70 %. Other results are overviewed and presented in Table 3. The experiment confirms that it is possible to evaluate the interoperability capability, i.e., identify the pairs of specific operations that potentially can be interoperable.

Table 3. Count of Operations with a given score for each software interoperability combination.

	Similarity >=		100 %				
	60%	70%	Enseble	Levenshtein	Jaro-Winkler	Jaccard	Longest Common Subsequence
ExactOnline X NMBRS	40	20	-	-	-	-	-
ExactOnline X Prestashop	54	18	5	5	5	5	5
ExactOnline X SuiteCRM	48	12	-	-	-	8	-
NMBRS X Prestashop	11	6	1	1	1	1	1
MMBRS X SuiteCRM	7	-	-	-	-	-	-
SuiteCRM X Prestashop	13	6	1	1	1	5	1

In Figure 8 the similarity of applications using different edit distance calculations is depicted. All edit distance algorithms determine the same similarity between the EAS (Fig. 8).

	ExactOnline	NMBRS	Prestashop	SuiteCRM
ExactOnline		17%	21%	15%
NMBRS	17%		16%	13%
Prestashop	21%	16%		14%
SuiteCRM	15%	13%	14%	

a)

	ExactOnline	NMBRS	Prestashop	SuiteCRM
ExactOnline		50%	53%	48%
NMBRS	50%		48%	46%
Prestashop	53%	48%		49%
SuiteCRM	48%	46%	49%	

b)

	ExactOnline	NMBRS	Prestashop	SuiteCRM
ExactOnline		3%	10%	7%
NMBRS	3%		2%	3%
Prestashop	10%	2%		7%
SuiteCRM	7%	3%	7%	

c)

	ExactOnline	NMBRS	Prestashop	SuiteCRM
ExactOnline		28%	33%	27%
NMBRS	28%		27%	24%
Prestashop	33%	27%		27%
SuiteCRM	27%	23%	27%	

d)

	ExactOnline	NMBRS	Prestashop	SuiteCRM
ExactOnline		24%	29%	24%
NMBRS	24%		23%	21%
Prestashop	29%	23%		24%
SuiteCRM	24%	21%	24%	

e)

Fig. 8. Similarity of applications using edit distance calculations a) Levenshtein, b) Jaro-Winkler, c) Jaccard, d) Longest common subsequence, e) ensemble.

The scoring amplitudes are somewhat shifted (a – [13; 21], b – [46;53], c – [2;10], d – [23;33], e – [21;29]) because of the difference of the edit distance calculation methods. The method can compare a different amount of procedures. The lower the percentage, the more procedures were compare, but the score was lower because of different amounts. It is still more important to check each comparison method rather than looking for a difference in each of them.

1.4. Interoperability evaluation using bag of words

Bag of words is a good model to simplify visualizations of data that was used in the experiment. In this research the bag of words method was used for data visualization and further decision making on experiment steps. We also used bag of words solution to split additive words such as “sendInvoice” so we could analyze separate words for example “send” and “invoice” separately. This helps determine that “send” is a verb and is used in action to the noun “invoice” which is an object in the application that is being analyzed. Determining and displaying bag of words helps visually see the application similarity results. An example is given using KonaKart, Zen Cart and Suite CRM application analysis in text analysis figures A, B and C (Fig. 9).

Fig. 9. Textual analysis comparison between applications using the bag of words method.

From the textual analysis comparison with the bag of words method we see the larger words that have most of related operations (Fig. 9) expressed. The operations that are verbs impact the objects which are nouns, and we can clearly see that KonaKart (A) and Zen Cart (B) have product an object that is possibly related. We can certainly say that (A) and (B) share same objects and therefore can be interoperable because we know that both applications are E-Commerce solutions and the method above gives us a computable objective view of the latter statement.

1.5. Interoperability evaluation using latent semantic analysis

The assumption that words in applications are semantically similar if they repeat in the similar places of the text – that is also known as distributed semantics. Based on this assumption, we can use the Latent Semantic Indexing (LSI) method to improve edit-distance method experiment results and to improve the monitoring and analysis actions of autonomic computing component. For latent semantic analysis we used R language version 3.5.1 and these libraries:

- RODBC – data reading and writing
- tm – a text mining tool
- quanteda – a text analysis tool with latent semantic analysis capability.

Experiment tests were carried out using the Latent Semantic Analysis tool from Quanteda library package. Latent semantic analysis is described in Information Retrieval, Algorithms and Heuristics book (Grossman & Ophir, 2012).

In the first experiment we compare ExactOnline and SuiteCRM applications, for only operations that are 100 percent match and try to see if they are similar by adding Objects, Fields, and Field Type information, hence the semantic knowledge about operation. We can clearly see that ExactOnline objects are more different from SuiteCRM package objects (Fig. 10)

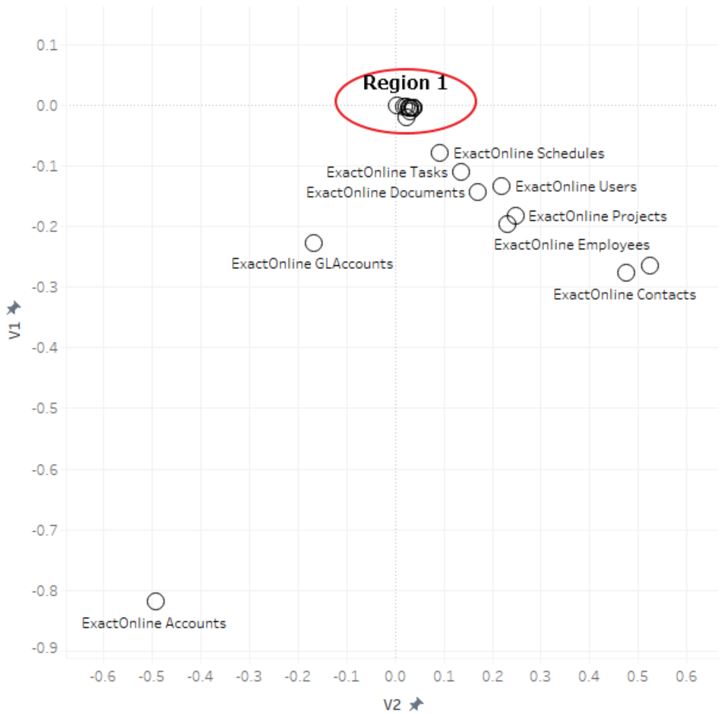

Fig. 10 ExactOnline comparison to SuiteCRM structural similarity using the LSI method.

From the ExactOnline comparison to the SuiteCRM figure it is seen that the vectors V1 and V2 reflect the positions of the objects on the plane (Fig. 10). The closer the objects in this plain are, the more related semantically they are, hence increasing a total possibility for applications to have interoperability.

6. CONCLUSIONS AND RECOMMENDATIONS

1. Most common problems of application integration and interoperability were listed and compared and showed that the most important problems are schema matching, orchestration and choreography in interoperability solutions.
2. Review of the currently existing interoperability evaluation methods show that they rely mostly on manual analysis, questionnaires and there are no automated approaches to determine whether multiple applications can be interoperable.
3. Main analyzed interoperability evaluation methods are LISI, I-Score, Comparison by functionality. The LISI and Comparison by functionality methods are quite similar, but LISI is more developed for a multiple business layer, and comparison by functionality deeply depends on the observer's subjective view of the domain. The I-score method is more technical and is closer to the topic of this research, but it covers only a very low technical level and does not deal with schema matching orchestration and choreography problems.
4. The proposed solution for autonomic interoperability evaluation was laid in the theoretical part of the dissertation. In the proposition it was argued that multiple knowledge sources of business domain can be used to add to the evaluation of interoperability. The proposed method suggests that knowledge can be gathered from business process, application architecture description files, and other ontology sources that could be added to the existing experiment and compared with target application, which would allow determining the coverage of the business layer to the application layer and how well CIM represents software PIM models in the enterprise architecture domain.
5. The presented experiment argues that the statement proposed method is able to autonomically detect similarity between applications by the highest-level using web service description documents and edit-distance, latent semantic analysis methods to get the quantitative evaluation of interoperability.

6. Enterprise applications were analyzed and evaluated the level of capability to be interoperable. The goal to assess interoperability through the knowledge available by automated algorithms has not yet been covered in the available solutions.
7. This research opens a possibility for a machine to machine interaction evaluation, helping people that work on integration projects.
8. Results of the current research might be helpful as decision support to quickly gain knowledge of compatibility between the systems.
9. In the experiment, 13 software systems were compared by difference edit-distance methods and give the output of evaluation of the capability of interoperability in the form of similarity score.
10. The negative side of such scoring is that the summary of API operation similarity score does not provide a full picture of similar objects and operation count difference in all applications and might affect this scoring method.
11. Jaccard, Jaro-Winkler, Levenshtein, and Longest Common Subsequence methods show the same separation of interoperability measure. The methods have a different level of precision estimating not such similar strings (below 60%).
12. Topic of the research could be expanded to investigate how autonomic component can evaluate interoperability when its managed application systems are not designed using service-oriented architecture.

This research provides the basis for supporting Business Process alignment to Application Processes and may impact the quality of application interoperability when using business process models. The idea is that after measuring whether software systems are interoperable, it is possible to measure the alignment to business processes and see which operation fall outside of the business process model.

LIST OF PUBLICATIONS ON THE TOPIC OF DISSERTATION

Chen, D., Doumeingts, G. & Vernadat, F., 2008. Architectures for enterprise integration and interoperability: Past, present and future. *Computers in industry,* 59(7), pp. 647-659.

Cintuglu, M. H., Youssef, T. & Mohammed, O. A., 2016. Development and application of a real-time testbed for multiagent system interoperability: A case study on hierarchical microgrid control. *IEEE Transactions on Smart Grid,* 9(3), pp. 1759-1768.

David, C., 2006. *Enterprise Interoperability Framework.* s.l., EMOI-INTEROP.

Dijkman, R. et al., 20144. Similarity of business process models: Metrics and evaluation.. *Information Systems,* 36(2), pp. 498-516.

Di, L. & Kobler, B., 2000. NASA standards for earth remote sensing data. *International Archives of Photogrammetry and Remote Sensing, 33(B2; PART 2),* pp. 147-155.

Dzemydienė, D. & Naujikienė, R., 2009. Elektroninių viešųjų paslaugų naudojimo ir informacinių sistemų sąveikumo vertinimas. *Informacijos mokslai,* Issue 50, pp. 267-273.

El-Halwagi, M. M., 2007. Process integration. *Elsevier,* Volume 7.

Fielding, R. T. & Taylor, R. N., 2000. *Architectural styles and the design of network-based software architectures.* Irvine: University of California.

Ford, T., Colombi, J., Graham, S. & Jacques, D., 2008. *Measuring system interoperability.* s.l., Proceeding Cser.

Gates, B., 2013. *Measuring progress. Annual Letter, Gates Foundation.* [Online] Available at: https://www.gatesfoundation.org/Who-We-Are/Resources-and-Media/Annual-Letters-List/Annual-Letter-2013 [Accessed 25 September 2018].

Gediminas, G., 2015. *Daugiaagentinių sistemų kūrimo metodų išvystymas nedidelio našumo įterptinių sistemų integravimui,* Vilnius: Vilnius University.

Gonzalo, N., 2001. A guided tour to approximate string matching.. *ACM computing surveys (CSUR),* 1(33), pp. 31-88.

Grossman, D. A. & Ophir, F., 2012. Information retrieval: Algorithms and heuristics. *pringer Science & Business Media,* Volume 15.

Hophe, G. & Woolf, B., 2004. *Enterprise integration patterns: Designing, building, and deploying messaging solutions.* s.l.:Addison-Wesley Professional.

Jackob, B., Lanyon-Hogg, R., Nadgir, D. & Yassin, A., 2004. *A practical guide to the IBM autonomic computing toolkit. IBM,* Durham: International Technical Support Organization.

Kasunic, M., 2001. *Measuring systems interoperability: Challenges and opportunities.* s.l.:Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

Kephart, J. O. & Chess, D. M., 2003. The vision of autonomic computing. *Computer,* Volume 1, pp. 41-50.

Krajicek, J. & Krajíček, J., 1995. *Bounded arithmetic, propositional logic and complexity theory*, Cambridge: Cambridge University Press.

Li, L., Wu, B. & Yang, Y., 2005. *Agent-based Ontology Integration for Ontology-based Applications.* Proceedings of the 2005 Australasian Ontology Workshop-Volume 58, Australian Computer Society, Inc..

McCann, R. et al., 2005. *Mapping maintenance for data integration systems.* s.l., VLDB Endowment, pp. 1018-1029.

Morkevičius, A., 2013. *Business and information systems alignment method based on enterprise architecture models,* Kaunas: KAUNAS UNIVERSITY OF TECHNOLOGY.

Overeinder, B. J. & Verkaik, P. D. B. F. M., 2008. Web service access management for integration with agent systems. *Proceedings of the 2008 ACM symposium on Applied computing,* pp. 1854-1860.

Pavlin, G., Kamermans, M. & Scafeş, M., 2009. Dynamic process integration framework: Toward efficient information processing in complex distributed systems. *Intelligent Distributed Computing III,* pp. 161-174.

Peukert, E., Eberius, J. & Rahm, E., 2012. *A self-configuring schema matching system.* s.l., In 2012 IEEE 28th International Conference on Data Engineering, pp. 306-317.

Rahm, E. & Bernstein, P. A., 2001. A survey of approaches to automatic schema matching. *The VLDB Journal,* 4(10), pp. 334-350.

Rezaei, R., Chiew, T. K. & Lee, S. P., 2014. A review on E-business Interoperability Frameworks. *Journal of Systems and Software,* Volume 93, pp. 199-216.

Shvaiko, P. & Euzenat, J., 2011. Ontology matching: state of the art and future challenges. *IEEE Transactions on knowledge and data engineering,* 1(25), pp. 158-176.

Silverston, L., 2011. *The data model resource book, Volume 1: A library of universal data models for all enterprises.*. s.l.:John Wiley & Sons.

Silverston, L., Inmon, W. H. & Graziano, K., 1997. *he data model resource book: a library of logical data models and data warehouse designs.* s.l.:John Wiley & Sons, Inc.

Tolk, A. & Muguira, J. A., 2003. *The levels of conceptual interoperability model.* s.l., Citeseer, pp. 1-11.

Trotta, G., 2003. Dancing around EAI 'bear traps'. *Business Process Management (BPM) Best Practices.*

Valatavičius, A. & Gudas, S., 2015. Enterprise software system integration using autonomic computing. *CEUR-WS,* Issue 1420, pp. 156-163.

Valatavičius, A. & Gudas, S., 2018. Measuring Enterprise Application Software Interoperability Capability. *CEUR Workshop Proceedings,* Volume 2158, pp. 104-113.

van der Bosch, M. A., van Steenbergen, M. E., Lamaitre, M. & Bos, R., 2010. A selection-method for Enterprise Application Integration solutions.. *nternational Conference on Business Informatics Research,* pp. 176-187.

Zinnikus, I., Hahn, C. & Fischer, K., 2008. A model-driven, agent-based approach for the integration of services into a collaborative business process.. *International Foundation for Autonomous Agents and Multiagent Systems,* Volume 1, pp. 241-248.

Левенштейн, В. И., 1965. Двоичные коды с исправлением выпадений, вставок и замещений символов. *Доклады Академии наук,* 163(4), pp. 845-848.

LIST OF PUBLICATIONS ON THE TOPIC OF DISSERTATION

The results of the research were published in two peer-reviewed journals:

1. Valatavičius, A., Gudas, S., Apie taikomųjų programų sąveikumo metodologiją, grindžiamą giluminėmis žiniomis. Informacijos mokslai, 79(79), pp.83-113.
2. Publikuotas straipsnis: Gudas, S., Valatavicius, A., 2017. Normalization of Domain Modeling in Enterprise Software Development. Baltic Journal of Modern Computing, 5(4), pp.329-350.

The results of the research were published in six peer-reviewed conference proceeding journals:

1. Valatavičius, Andrius & Gudas, Saulius, 2015. Towards business process integration using autonomic computing. Informacinės technologijos 2015: Konferencijos pranešimų medžiaga, pp.81–84.
2. Valatavičius, A. and Gudas, S., Apie taikomųjų programų sąveikumo metodologiją, grindžiamą giluminėmis žiniomis. Informacijos mokslai, 79(79), pp.83-113.
3. Valatavičius, A. and Gudas, S., 2018. Measuring Enterprise Application Software Interoperability Capability.
4. Valatavičius, A. and Gudas, S., 2015. Enterprise software system integration using autonomic computing. *CEUR-WS. org*, *1420*, pp.156-163.
5. Valatavičius, Andrius & Gudas, Saulius, 2016. Modeling environment to maintain interoperability of enterprise applications. Data analysis methods for software systems: 8th international workshop on data analysis methods for software systems, Druskininkai, December 1-3, 2016, pp.63–64.
6. Valatavičius, Andrius & Gudas, Saulius, 2017. Advanced evaluation methods of multiple application software interoperability. 9th International workshop on Data Analysis Methods for Software Systems (DAMSS), Druskininkai, Lithuania, November 30 - December 2, 2017, p.52.
7. Valatavičius, Andrius & Gudas, Saulius, 2018. Advanced evaluation methods of multiple application software

interoperability. 10th International workshop on Data Analysis Methods for Software Systems (DAMSS), Druskininkai, Lithuania, November 29 - December 1, 2018, p.87.

The resulst of the research were presented at six national and international conferences:

1. Tarptautinė konferencija: Dalyvauta Doktorantų konsorciume BIR 2015 Estijoje tema: “Enterprise Software System Integration using Autonomic Computing“;
2. Tarptautinė konferencija: DB&IS 2016 Latvijoje tema: “Modelling Dynamic Enterprise Environment to Maintain Interoperability of Applications“;
3. Tarptautinė konferencija: DAMSS „Data Analysis Methods for Software Systems“ 2016;
4. Konferencija: XVIII tarptautinėje kompiuterininkų konferencijoje LIKS 2017, tema: Towards deep knowledge based interoperability of applications. Straipsnis priimtas publikacijai žurnale „Informacijos Mokslai“;

ABOUT THE AUTHOR

Andrius Valatavičius obtained BSc degree in 2012 in the field of Business Informatics and MCs degree in 2014 in the field of Business Informatics, both at Vilnius University, Kaunas faculty of humanities. He was a PhD student at Vilnius University Institute of Data Science and Digital Technologies from 2014 to 2018. Currently he is a freelancing data architecture specialist working on multiple projects with UAB “Kvantas” and UAB “Intellerts”. His interests include enteprise application integration and interoperability methods and tools.

NOTES

NOTES

Andrius Valatavičius

ENTERPRISE APPLICATION INTEROPERABILITY EVALUATION USING AUTONOMIC COMPUTING

Summary of a Doctoral Dissertation
Natural Sciences
Informatics (N 009)
Editor Zuzana Šiušaitė

Andrius Valatavičius

TAIKOMŲJŲ PROGRAMŲ SĄVEIKUMO VERTINIMAS TAIKANT AUTONOMINIO SKAIČIAVIMO TECHNOLOGIJAS

Daktaro disertacijos santrauka
Gamtos mokslai
Informatika (N 009)
Redaktorė Jorūnė Rimeisytė - Nekrašienė

Vilniaus universiteto leidykla
Saulėtekio al. 9, LT-10222 Vilnius
El. p. info@leidykla.vu.lt,
www.leidykla.vu.lt
Tiražas 30 egz.