JVLC

Journal of Visual Language and Computing

Volume 2020, Number 2

DOI: 10.18293/JVLC2020-N2

Journal preparation, editing and printing are sponsored by KSI Research Inc.

Journal of
Visual Language and Computing

An Open Access Journal published by

KSI Research Inc.

156 Park Square Lane, Pittsburgh, PA 15238 USA

Journal of
Visual Language and Computing

Volume 2020, Number 2

December 2020

Table of Contents

Regular Papers

Research Notes

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc/

A Multilayer Graph Approach for Predicting Computer Network Cyber-attacks

Francesco Colace [a], Muhammad Khan [b], Marco Lombardi [a], Domenico Santaniello [a, *]

[a]DIIn University of Salerno, Italy

[a]New York University, Abu Dhabi, United Arab Emirates

ARTICLE INFO

Article History:
Submitted 8.18.2020
Revised 8.25.2020
Second Revision 10.20.2020
Accepted 11.30.2020

Keywords:
Network Security
Knowledge Management
Bayesian Network
Probabilistic Graphical Models

ABSTRACT

Today's society is heavily oriented towards digitalization, which increasingly affects the management of cities and services. This process is performed through the use of the Internet of Things (IoT) paradigm, from which arise problems related to security. In this scenario, based on the continuous exchange of information on the network, an increasingly significant role is played by systems able to guarantee data security. Protecting the modern Computer Networks could be a very complex task. In this paper, a methodology based on three graphic models (Context Dimension Tree, Ontology and Bayesian Network) is proposed. Three different models are used which use context representation and probabilistic approaches to predict cyber-attacks. The paper proposes, in fact, the use of Bayesian networks built through an ontological definition of the problem dropped on a certain context represented by a Context Dimension Tree. The proposed approach has been experimented in a real scenario providing satisfactory results.

1. Introduction

Modern digitization allowed the development of increasingly smart environments capable of managing countless services designed for citizens. Nowadays, many services designed to improve users' activities are made available with the use of modern devices. This process has been made possible through the Internet of Things (IoT) paradigm [1], which represents a concept where objects and users are interconnected and exchange information through the Internet [2]. One of the particularly interesting issues of this scenario is represented by systems able to guarantee network security [3]. In particular, the security is increased through systems designed to control the network such as Intrusion Detection Systems (IDSs). IDSs are systems capable of analyzing every packet which is exchanged on the network. Those packets, containing the exchanged information, may contain possible threats that can compromise the entire network. In attempting to successfully identify cyber-attacks, these systems work through a database containing a set of rules used to identify security violations, basically comparing the content of the packets with known violation rules. However, this approach remains completely vulnerable to possible new types of attacks and cannot predict what may occur in the immediate future. Therefore, a further study of systems able to identify network attacks based not only on comparison but also on data behavior is needed.

The aim of this paper is to propose a methodology capable of recognizing and dealing with problems related to cyber-attacks, ensuring network security. The proposed methodology exploits different graphic formalisms (Ontology, Context Dimension Tree) able to represent and identify problems related to security. This approach is based not only on the comparison of known threats but also on the behavior of data on the network trying to predict potential cyber-attacks.

The paper is organized as follows: section two offers a general overview of the problem of network security with reference to related works; section three shows the proposed methodology; section four evaluates the performance of the system presented through a case study application. The article ends with the conclusions.

*Corresponding author
Email address: dsantaniello@unisa.it
Website: http://docenti.unisa.it/domenico.santaniello
ORCID: 0000-0002-5783-1847

DOI reference number: 10.18293/JVLC2020N2-006

2. Background

Internet has become a very important tool for institutions such as businesses, universities and public administration. Beyond this, the modern human being relies on the internet in many social and personal professional activities. Over the years, this type of use has given particular attention to the field of information security, in particular, the field of network security is concerned with defending networks from possible attacks, being able to recognize and classify them in order to mitigate risks that they involve. When we connect to the computer network during all our daily activities, we do it for the purpose of exchanging information. This operation, in electronic language, translates into packet exchange; however, these packets may contain malicious content that we identify with the name of malware. These malicious packets aim to establish themselves in our computer devices, extorting sensitive information and threatening the safety of the entire computer network to which the device be-longs [4]. Some of these use self-replicating technologies, therefore able to self-replicate indefinitely within a system by sending its replicas with the attempt to infect the whole system, in some cases, these malwares are designed to act without establishing any kind of explicit interaction with the user (worms). For this reason, it is important to prevent and protect not only users but also computer networks. A broad category of security threats falls into the Denial of Service (DoS) class; such attacks aim to render an IT service unusable, which, as we said earlier, can be crucial with respect to the formal fulfilment of imprints, university or public administration, which the system is called upon to perform. These types of attacks, in general, can be classified into three categories: 1) attacks on the system vulnerability, which involves sending packets to the most vulnerable system within the network; 2) Band Flooding, which involves sending a deluge of packets with the aim of obstructing the connection to the service; 3) Connection Flooding which aims to establish a large number of connections that keep the system busy, preventing connections to be established to users requesting services. Furthermore, these types of DoS attacks can be effective exploiting multiple or distributed sources, thus speaking of DDoS, increasing their danger and decreasing the possibility of detection and blocking.

In literature, several papers deal with the problem of network security in the Internet of Things or Smart City fields ([5]–[7]). A common approach is to introduce of a Framework able to analyse networks trying to manage any attacks or disruption.

Elsaeidy, in this work [8], proposes an interesting approach based on Deep Learning and user data behaviour. The aim of the approach is based on recognitions of patterns, which could predict potential cyber-attacks. In [9] is proposed an approach based on artificial intelligence techniques applied in industrial sector in order to identify any network problems or threat. The approach takes advantage of attacks trees in order to identify and design defence strategies. Moreover, in [10] the Hybrid Attack Graph (HAG) is presented, which model and combines physical and software component of attacks necessary for potential risks picture overview. Furthermore, promising results are provided in [11] where Data Mining techniques (Multilayer Perceptron, Naive Bayes and Random Forest) are used to determine the type of attack.

Starting from this general overview, and taking advantage of the literature, it seems to be possible to design an approach able to integrate semantic, probabilistic and context aware approaches in a single methodology. The approach proposed in this paper introduces a multilevel graph approach based con Ontology, Bayesian Network and Context Dimension Tree able to perform a complete and detailed analysis of the network status. Taking as reference the following articles of literature [12], [13] it was possible to create a networked ontology of security suitable for the specific problem under consideration. This tool allows us therefore to have a detailed classification of the problem with all the possible relations capable of generating the inferences useful to our approach.

Although the tree graph approach is not new in the application of this field [9], [11], the design of a specific CDT represents a novel approach to the problem. This tool is able to represent and manage all the possible contexts of the application do-main dealt with. Moreover, thanks to the combined use with ontology it is possible to apply the proposed methodology.

2.1 Intrusion Detection Systems

According to the previous paragraph, to make a computer network secure, we need to check all the packets we have exchanged. We therefore need a device that not only examines packet headers (such as a firewall, for example) but also performs a thorough and detailed check of the packet. Intrusion detection system (IDS), these systems are suspect-driven. Through them, it is possible, possibly, to prevent access to these packets [14]. These systems are used to detect a wide range of attacks, including network mapping, port scans, DoS and DDoS attacks, worms and viruses, application vulnerability attacks. Today thousands of organizations exploit this type of system in their institutional networks, acting as sensors that work together exchanging information between each other and communicating to the network administrator any suspicious activity. In general, IDS systems are classified as signature-based or anomaly-based systems. IDS based on signatures, has a database of attack signatures, which represents the set of rules concerning an intrusion activity. Operationally, this type of system checks each packet that passes through it, comparing it with the signatures in its database.

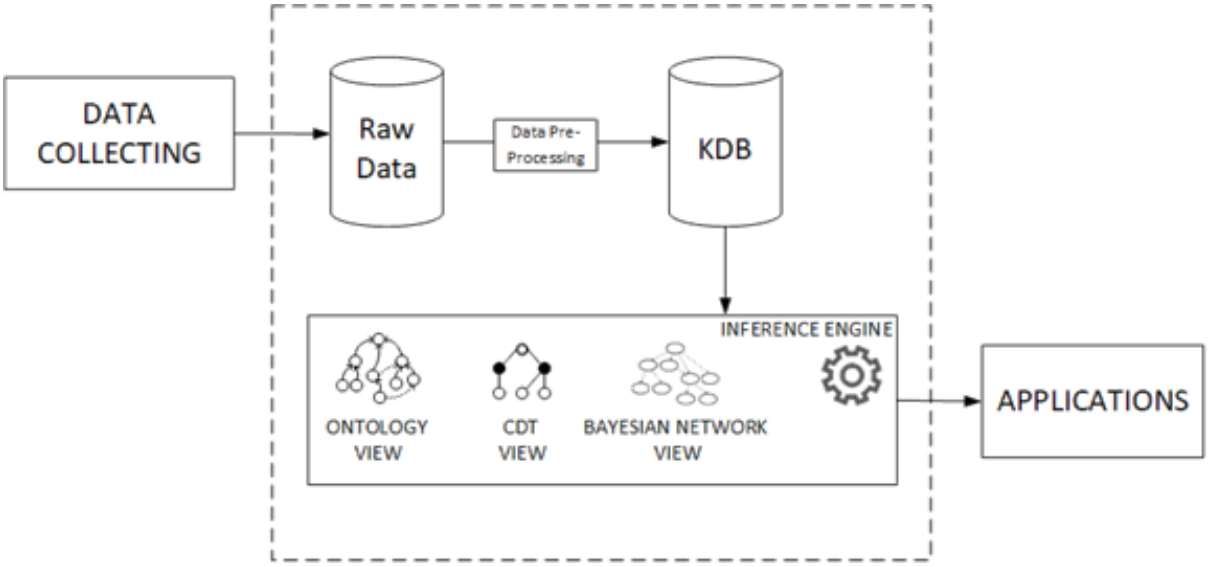


Figure 1: The System Architecture.

IDS based on anomalies, on the other hand, collects traffic information trying to find anomalous flows. In light of this, the study of a methodology that could assist these security systems, thus able to recognize possible threats, suggest possible mitigation interventions and possibly foresee such phenomena based on the context, becomes necessary to protect the computer networks and their users.

3. The Proposed Approach

In consideration of the preceding points, the proposed methodology is designed to be a predictive approach capable of adapting to the context. This approach is useful in various fields [15]–[17]. In particular, this article presents an application of the proposed methodology in the field of cyber-attacks. Three graph approaches such as Ontology, CDT and Bayesian Network are exploited to detect and predict the occurrence of events malicious to the network that would compromise the service and security for users. Bayesian Networks are particularly useful in the attempt to predict specific events [18], moreover, they are able to interface adequately with other graph approaches. The CDT is a tree able to manage and customize information present in all possible contexts [19]. Furthermore, Ontologies are used for the representation of reality, being particularly useful and interfaceable with the other two graph approaches used [20]–[22].

According to the proposed approach, the two graph approaches, responsible for the representation of the context (CDT and Ontology), can be combined in order to obtain a list of constraints useful for the design of the Bayesian Networks. In detail, a recognition of all possible context combinations can be made through the CDT. These combinations of contexts are useful for extracting relationships through the various nodes represented through the Ontology view. The relationships extracted from the Ontological view can be transformed into useful constraints for the construction of the structure of the Bayesian Network, which will be improved through knowledge of the context.

The figure 1 shows the system architecture, which, by collecting raw data, uses them to return the appropriate usage application. In the first phase there is the collection of data from the IDS and other sensors allocated in the computer network, which are stored raw. These data are harmonized and sorted in the pre-processing phase and stored in a database that powers the inference engine. Inside the inferential engine are the three graph views previously described (CDTs, Ontologies and Bayesian Networks) which provide an interpretation of the knowledge acquired and collected in Knowledge Database (KDB).

In practice, the detection of malicious attacks, given a certain context, could be done through the described architecture that exploits the right information characterized by innovative elements based on formal context representation, knowledge management organization and inferential engines.

Information management systems require particular attention to the efficiency of data organization. In this regard, Ontologies represent a particularly suitable means for the organization and reuse of shared and collaborative knowledge, bringing advantages in terms of overall system efficiency [23]. The proposed system makes use of Context Awareness, wants to be able to manage real-time contextual information that could bring improvements in the identification and predictive capacity of the system. The CDT is a tree with the ability to represents all possible contexts, composed of a root node, a set of leaf nodes interconnected each other. The CDT's nodes are divided in 1) Black Nodes,

which represents the dimension nodes of the domain; 2) White Nodes or concept nodes, which contains all dimensions values. A Context is specified as an "and" among different context elements where each element is defined as an assignment dimension_namei = value: several context elements, combined with each other, give rise to a context [24]. The proposed approach, moreover, takes advantage of probabilistic approach through Bayesian Networks, which are used to compute and update the probability of a given event taking advantage of Bayes' Theorem.

4. Experimental Results

The purpose of this section is to illustrate the experimental application of the proposed methodology to a real case. The methodology proposed, as previously mentioned, combines three approaches to graph (CDT, Ontology and BN) in order to provide answers in terms of forecasting events; in particular, in this experimental case the events refer to network attacks. The proposed approach is able to combine the CDT and Ontology views to extrapolate the conceptual relationships, these relationships are transformed into constraints and are used to service the construction of the Bayesian Network. By means of this graph, which contains all the relations and the relative weights between the available data, it is possible to predict the occurrence of an event as the boundary conditions vary.

The experimental phase was conducted through the use of a dataset containing data from Intrusion Detection Systems at the service of a university computer network. The dataset contains over twenty thousand instances and represents a reduced set of the monitoring database at service since 2008.In particular, the dataset used contains over two thousand attempts to attack, of which only about 5% succeeded in penetrate the computer network. To perform the analysis of the proposed approach it was necessary to divide the dataset into training set (90%) and testing set (10%), this subdivision was made taking into account a balance between number and type of events and periods of intense information communication technologies activities. The training set is fundamental during the learning phase of the network structure, instead, through the test set, the network is validated verifying if it is able to correctly classify the events present in the test set. For the purposes of validating the model, an automatic network learning algorithm was chosen, and a comparison was made between two cases. The first case, which involves the use of the learning algorithm only to define the network structure, and the second case that use the proposed methodology, which combines the same automatic learning algorithm with a list of constraints coming from the combination of CDT and the Ontological view as described above. The machine-learning algorithm chosen is Hill Climbing with score K2 [25], which, among the various selected algorithms, has provided greater feedback in terms of Overall Accuracy in reference to our case study. The results obtained from the experimental case are shown in terms of Overall Confusion Matrix Accuracy Precision and Recall. The confusion matrix is a useful tool for representing the accuracy of statistical classification, through this matrix it is possible to have an overall view of the classification ability of the Bayesian Network built, furthermore it is possible to calculate several coefficients that help us to understand reliability of the Bayes Network. The coefficients used are the Accuracy which represents the proportion of events correctly classified with respect to all events, the Precision which can be seen as a measure of accuracy or fidelity of the forecasts and the Recall which can be seen as a measure of completeness.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

The analysis was carried out on a dataset of data from Intrusion Detection System monitoring a university network. Among the many data available, the following types of network attacks were selected:

- Denial of Service (DoS)
- Distributed Denial of Service (DDoS)
- Spear Phishing (SPh)
- Web Deface (WD)
- Password Harvesting (PH)

The aim of this experimentation phase, therefore, was to foresee such attacks, first through a network trained through a selected learning algorithm, and subsequently using the proposed methodology.

As shown in Table 1, the confusion matrix of the first case refers to the matrix learned by means of the selected learning algorithm. Through this algorithm, it was possible to obtain a network, which is able to correctly classify some events, in particular many DoS attacks. The obtained matrix does not show excellent results in terms of Overall Accuracy (Table 1) and in terms of Precision and Recall (Table 3).

Table 1: Confusion Matrix Case1.

		Reference				
		DDoS	DoS	WD	SPh	PH
Prediction	DDoS	189	87	25	21	79
	DoS	77	254	14	65	78
	WD	23	94	280	46	45
	SPh	45	32	42	172	88
	PH	13	96	42	31	209

Overall Accuracy : 51,42%

The Table 2 shows the confusion matrix obtained from the Bayesian Network Structure designed by the proposed approach. In fact, compared to previous case

(Table1), there is an increasing of the number of correctly classified events and a decreasing of incorrectly classified events.

Table 2: Confusion Matrix Case2.

		Reference				
		DDoS	DoS	WD	SPh	PH
Prediction	DDoS	358	62	14	11	37
	DoS	57	419	20	18	36
	WD	18	41	348	39	24
	SPh	33	12	24	216	19
	PH	7	18	4	23	289

Overall Accuracy : 75,92%

This improvement can be seen in the increase of Overall Accuracy, which exceeds 75%, and as witnessed in Table 3 a significant increase compared to the previous case in terms of Precision and Recall.

Table 3: Precision and Recall Parameters Case1 and Case2.

		DDoS	DoS	WD	SPh	PH
Case 1	Precision	47,13%	52,05%	57,38%	45,38%	53,45%
	Recall	54,47%	45,11%	69,48%	51,34%	41,89%
Case 2	Precision	74,27%	76,18%	74,04%	71,05%	84,75%
	Recall	75,69%	75,91%	84,88%	70,36%	71,36%

The value of about 76% of Overall Accuracy may seem a reasonable result, nevertheless, compared to the performance of modern machine learning algorithms, it is not a great result in absolute terms of forecasting. However, another aspect related to the proposed methodology can be analysed, which can be fundamental in the attempt to classify the attacks that actually have been successful. This aspect was analysed by testing the Bayesian Network, which was learned through the use of the proposed methodology, through an ad hoc test dataset, which contains only the cyber-attacks that penetrated the computer network. In this case, as can be seen in Figure 2, the system was able to correctly classify over 60% of the events. The result obtained was achieved through the system's ability to understand the context by classifying events, unlike traditional control tools, according to data behaviour. This aspect suggests to us that the capacity of the adopted methodology can be fundamental in further reducing the percentage of network attacks that have been successful in the attack of the networks, intervening especially where the traditional control system does not recognize the packets as possible attacks.

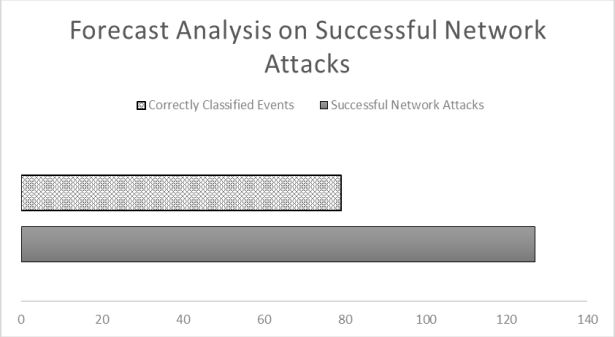


Figure 2: Forecast Analysis on Successful Network Attacks.

5. Conclusions

This paper aimed to introduce and analyse the performance of a multi-level graph methodology for cyber-attacks predictions. To perform the analysis was used a dataset of Intrusion Detection Systems, which monitor the computer network at the service of a university institution. The dataset includes over two thousand attack attempts, of which about 5% defeat the computer network security systems. The analysis was conducted using part of dataset to build the Bayesian Network structure, which was tested with the remaining part of data. Therefore, was compared the performance of the Bayesian Network structure built through a machine learning algorithm and the Bayesian Network structure built through the proposed methodology. Furthermore, is evaluated the performance of Bayesian Network structure learned through the proposed methodology in predict the cyber-attack events that defeated the security systems of the networks.

According to the confusion matrices (Table 1 and Table 2) and the results in terms of Prediction and Recall (Table 3), the proposed system, compared to a traditional structural learning algorithm, has been able to provide good performance in terms of Overall Accuracy, Prediction and Recall. The results seem not enough in absolute classification terms; however, the system strength lies in using graph approaches, which provide a better description of the problem allowing prediction and classifying of events based on data behaviour. In fact, the structure of the network learned in the second case, that is through our approach, despite having obtained only about 76% of accuracy, when it was tested with a dataset containing only the attacks on the computer network that actually penetrated the system, is was able to correctly classify over 60% of these events (Figure 2).

From the analysis of the experimental data, it is clear that the proposed system does not want to a replacement of the modern Intrusion Detection Systems but can be adequately able to support them. In particular, the capacity of the proposed system lies in intervening against unknown cyber-attacks that could compromise the security of the computer network. The main advantages that can lead the proposed system to

improve its efficiency are two: the amount of data and the use of graph formalisms. In particular, as the number of data increases, the system is able to build Bayesian Network structures more reliable and able to provide more accurate results. The use of graph formalisms of the proposed approach, such as Ontologies, enable our system to communicate with other similar systems, exchanging useful information and knowledge that could lead the system to a continuous improvement.

References

[1] K. Ashton, "That 'Internet of Things' Thing," *RFiD J.*, 2009 DOI:10.1016/j.amjcard.2013.11.014.

[2] M. Carratu, M. Ferro, A. Pietrosanto, P. Sommella, and V. Paciello, "A Smart Wireless Sensor Network for PM10 Measurement," in *2019 IEEE International Symposium on Measurements and Networking, M and N 2019 - Proceedings*, 2019 DOI:10.1109/IWMN.2019.8805015.

[3] A. Castiglione, F. Palmieri, F. Colace, M. Lombardi, D. Santaniello, and G. D'Aniello, "Securing the internet of vehicles through lightweight block ciphers," *Pattern Recognit. Lett.*, 2020 DOI:10.1016/j.patrec.2020.04.038.

[4] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security)*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[5] A. Essa, T. Al-Shoura, A. Al Nabulsi, A. R. Al-Ali, and F. Aloul, "Cyber Physical Sensors System Security: Threats, Vulnerabilities, and Solutions," in *2018 2nd International Conference on Smart Grid and Smart Cities (ICSGSC)*, 2018, pp. 62–67 DOI:10.1109/ICSGSC.2018.8541316.

[6] S. Chakrabarty and D. W. Engels, "A secure IoT architecture for Smart Cities," in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2016, pp. 812–813 DOI:10.1109/CCNC.2016.7444889.

[7] J. Pacheco and S. Hariri, "IoT Security Framework for Smart Cyber Infrastructures," in *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016, pp. 242–247 DOI:10.1109/FAS-W.2016.58.

[8] [A. Elsaeidy, I. Elgendi, K. S. Munasinghe, D. Sharma, and A. Jamalipour, "A smart city cyber security platform for narrowband networks," in *2017 27th International Telecommunication Networks and Applications Conference, ITNAC 2017*, 2017, vol. 2017-Janua, pp. 1–6 DOI:10.1109/ATNAC.2017.8215388.

[9] G. Falco, A. Viswanathan, C. Caldera, and H. Shrobe, "A Master Attack Methodology for an AI-Based Automated Attack Planner for Smart Cities," *IEEE Access*, vol. 6, pp. 48360–48373, 2018 DOI:10.1109/ACCESS.2018.2867556.

[10] P. J. Hawrylak, M. Haney, M. Papa, and J. Hale, "Using hybrid attack graphs to model cyber-physical attacks in the Smart Grid," in *Proceedings - 2012 5th International Symposium on Resilient Control Systems, ISRCS 2012*, 2012, pp. 161–164 DOI:10.1109/ISRCS.2012.6309311.

[11] M. Alkasassbeh, G. Al-Naymat, A. B. Hassanat, and M. Almseidin, "Detecting distributed denial of service attacks using data mining techniques," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 1, 2016.

[12] M. Iannacone, S. Bohn, G. Nakamura, J. Gerth, K. Huffer, R. Bridges, E. Ferragut, and J. Goodall, "Developing an Ontology for Cyber Security Knowledge Graphs," in *Proceedings of the 10th Annual Cyber and Information Security Research Conference on - CISR '15*, 2015, pp. 1–4 DOI:10.1145/2746266.2746278.

[13] A. Oltramari, L. F. Cranor, R. J. Walls, and P. McDaniel, "Building an ontology of cyber security," in *CEUR Workshop Proceedings*, 2014.

[14] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, Jan. 2013 DOI:10.1016/J.JNCA.2012.09.004.

[15] F. Clarizia, F. Colace, M. De Santo, M. Lombardi, F. Pascale, D. Santaniello, and A. Tuker, "A multilevel graph approach for rainfall forecasting: A preliminary study case on London area," *Concurr. Comput. Pract. Exp.*, vol. 32, no. 8, Apr. 2020 DOI:10.1002/cpe.5289.

[16] F. Colace, M. Lombardi, F. Pascale, D. Santaniello, A. Tucker, and P. Villani, "MuG : A Multilevel Graph Representation for Big Data Interpretation," in *IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems*, 2018, pp. 1410–1415 DOI:10.1109/HPCC/SmartCity/DSS.2018.00233.

[17] F. Clarizia, F. Colace, M. Lombardi, F. Pascale, and D. Santaniello, "A Multilevel Graph Approach for Road Accidents Data Interpretation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11161 LNCS, 2018, pp. 303–316 DOI:10.1007/978-3-030-01689-0_24.

[18] P. Weber, G. Medina-Oliva, C. Simon, and B. Iung, "Overview on Bayesian networks applications for dependability, risk analysis and maintenance areas," *Engineering Applications of Artificial Intelligence*. 2012 DOI:10.1016/j.engappai.2010.06.002.

[19] M. Casillo, F. Clarizia, G. D'Aniello, M. De Santo, M. Lombardi, and D. Santaniello, "CHAT-Bot: a Cultural Heritage Aware Teller-Bot for supporting touristic experiences," *Pattern Recognit. Lett.*, Jan. 2020 DOI:10.1016/j.patrec.2020.01.003.

[20] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive computing environments," *Knowl. Eng. Rev.*, vol. 18, no. 3, pp. 197–207, 2003.

[21] E. M. Helsper and L. C. Van Der Gaag, "Building Bayesian networks through ontologies," *ECAI2002, Proc. 15th Eur. Conf. Artif. Intell.*, pp. 680–684, 2002.

[22] F. Colace and M. De Santo, "Ontology for E-learning: A Bayesian approach," *IEEE Trans. Educ.*, vol. 53, no. 2, pp. 223–233, 2010.

[23] F. Colace, M. Lombardi, F. Pascale, and D. Santaniello, "A Multilevel Graph Representation for Big Data Interpretation in Real Scenarios," in *Proceedings - 2018 3rd International Conference on System Reliability and Safety, ICSRS 2018*, 2019 DOI:10.1109/ICSRS.2018.8688834.

[24] M. Casillo, F. Clarizia, F. Colace, M. Lombardi, F. Pascale, and D. Santaniello, "An Approach for Recommending Contextualized Services in e-Tourism," *Information*, vol. 10, no. 5, p. 180, May 2019 DOI:10.3390/info10050180.

[25] G. F. Cooper and E. Herskovits, "A Bayesian Method for the Induction of Probabilistic Networks from Data," *Mach. Learn.*, 1992 DOI:10.1023/A:1022649401552.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc

Monitoring Evolution of Dependency Discovery Results

Loredana Caruccio, Stefano Cirillo

University of Salerno, via Giovanni Paolo II, 132 84084 Fisciano (SA), Italy

ARTICLE INFO

Article History:
Submitted 15.10.2020
Revised 11.29.2020
Accepted 12.4.2020

Keywords:
Data stream profiling
Big data visualization
Metadata visualization
Continuous discovery
Relaxed functional dependencies

ABSTRACT

The automatic discovery from data of Functional Dependencies (FDs), and their extensions Relaxed Functional Dependencies (RFDs), represents one of the main tasks in the data profiling research area. Several algorithms that deal with the "complex" problem of discovering RFDs have been recognized as a fundamental tool to automatically collect them starting from data. Moreover, the characteristics of scenarios involving "big" data require also profiling tasks to evolve towards continuous ones, which must be capable to dynamically collect and update the set of holding RFDs on the analyzed data. In this context, one of the most critical scenarios is represented by the possibility to discover RFDs over data streams. Nevertheless, although the main goal of discovery algorithms is allowing for fast execution processes, to enable the analysis of the resulting RFDs, it is necessary to also devise methods to continuously monitor discovery results. Thus, one of the main goals is to reduce the users' effort in moving in and out the possible huge quantity of holding RFDs. To this end, in this paper, we present DEVICE, a tool for continuously monitoring resulting RFDs during the execution of discovery processes. In particular, it permits to analyze the evolution of results by using a lattice representation of the search space. Moreover, zooming and filtering functionalities enable the user to focus the analysis on a specific portion of the search space. The effectiveness of the proposed tool has been evaluated in a scenario studying the application of different discovery strategies over a well-known and real-world dataset.

1. Introduction

Collecting metadata from big datasets is the goal of the data profiling research area, in which the discovery of functional dependencies (FDs), and their extensions relaxed functional dependencies (RFDs) represents one of its fundamental tasks. This kind of semantic property describes relationships among database attributes, which might be exploited in several advanced database operations, such as query optimization, data cleaning, and so forth. In particular, RFDs relax some constraints of canonical FDs by admitting the possibility for a dependency to hold on a subset of data (also referred to as RFDs relaxing on the *extent*), and/or by relying on approximate paradigms to compare pairs of tuples (also referred to as RFDs relaxing on the *attribute comparison*) [7].

Although the problem of discovering FDs and RFDs from data is extremely complex, the recent definition of efficient algorithms enabled their discovery from "big" datasets. Among these, it is worth to mention [20, 26, 22] for FD discovery, and [15, 25, 4, 7] for RFD discovery. Moreover, recent proposals dealt with incremental or continuous data profiling scenarios [23, 3, 2]. The latter are particularly useful in current application scenarios, such as big data analytic tasks, in which the possibility to learn predictive models from data requires to dynamically profile data streams and learn models from them. To this end, it is necessary to devise methods and tools capable of visualizing the dynamic evolution of the discovery results characterizing the profile of a data stream, and/or of the predictive models of interest. Indeed, a proper analysis of how RFDs change over time cannot be accomplished by looking at such a huge number of holding RFDs that change very rapidly.

We particularly focus on such kind of scenarios, where the goal is to get holding RFDs even when the input data dynamically change over time, permitting the discovery of RFDs also from data streams. The latter scenario imposes different emerging research challenges, such as the necessity of enabling user i) to continuously monitor and rapidly visualize discovery results, ii) to analyze how a discovery process browses the search space according to data are get-

lcaruccio@unisa.it (L. Caruccio); scirillo@unisa.it (S. Cirillo)
ORCID(s):

DOI reference number: 10.18293/JVLC2020N2-007

ting in, and iii) to easily interact with specific portions of the search space, entailing the focus on a specific portion of results.

To this end, in this paper, we present DEVICE (DEpendencies VIsualizer on lattiCE), which permits to monitor the set of RFDs extracted from data streams through a lattice representation. Moreover, DEVICE enables the user to interact with the discovery results by zooming on the search space and/or filtering results according to specific attributes. Finally, results can be also filtered according to RFD threshold settings. From an architectural point-of-view, DEVICE is scalable towards all possible (incremental) RFD discovery algorithms, also thanks to an input driver connector module devoted to the parsing of input data, so enabling the standardization of discovery results.

The paper is organized as follows. Section 2 describes related visualization approaches and tools. Section 3 presents the theoretical foundations of RFDs and the representation used to model the search space of the discovery problem. Section 4 presents DEVICE, whereas Section 5 reports an experimental case study on the application of different discovery strategies over a real-world dataset. Finally, summary and future directions are included in Section 6.

2. Related Work

Big data visualization is an important research area in which the main goal is to provide effective visualization techniques capable to describe data into their "big" and challenging contexts [21, 13]. In fact, not only the dynamic nature of data increases the complexity of design choices, but also the necessity to provide insights to users in real-time and to enable effective interactions with graphical components.

More specifically, there are several contexts in which it is important to improve the understanding of algorithm/model results and characteristics, such as the data mining [12, 11], data privacy [10, 17], and the deep learning [8]. To this end, in the literature, many approaches and tools have been proposed. Among these, it is worth to mention Association Rules (ARs) visualization approaches, since the concept of AR is somehow related to that of RFD. Effective examples are i) the tool in [24], which provides multiple views to visually inspect the overall set of ARs, and ii) the hierarchical matrix-based visualization technique presented in [9].

Concerning the discovery of RFDs, the availability of efficient RFD discovery algorithms yields the necessity to manage big result sets of RFDs, most of which differ only for some values of relaxation parameters. However, recently such algorithms are becoming capable to scale over big data sources, but there are no many solutions in the literature for handling the complexity related to the visualization of a possible huge number of discovered RFDs. Among these, one of the most effective platforms for data profiling is the Metanome project [18], which embeds several algorithms to automatically discover complex metadata, including functional and inclusion dependencies. Moreover, it embeds various result management techniques, such as list-based ranking techniques, and interactive diagrams of discovery results. Another representative scalable platform for analyzing data profiles is Metacrate [14], which permits the storage of different meta-data and their integrations, enabling users to perform several ad-hoc analysis. In this context, the first proposal for visualizing large sets of RFDs is described in [6]. It presents several metaphors for representing RFDs at different levels of detail. Starting from a high-level visualization of attribute correlations, details are interactively revealed, also including details on the relaxation criteria.

Although all of the above approaches represent effective tools to visualize and explore properties and metadata after the execution of mining/discovery algorithms, a recent proposal goes beyond the only result visualization problem [1], since it allows users to explore how RFDs change over time, and to perform result comparisons among different time-slots. The latter approach shares similar goals to our proposal. Nevertheless, it is mainly focused on the analysis of temporal trends related to the number of discovered RFDs. Instead, DEVICE is able to represent how discovery results change into the search space. This characteristic makes it also useful for the analysis of how different algorithms browse the search space.

3. Background

Before describing the proposed tool we will review some background definitions on the concept of RFD and the graph lattice representation.

A relational database schema $\mathcal{R}$ is defined as a collection of relation schemas $(R_1,\ldots, R_n)$, where each R_i is defined over a set $attr(R_i)$ of attributes $(A_1,\ldots, A_m)$. Each attribute A_k has associated a domain $dom(A_k)$, which can be finite or infinite. A relation instance (or simply a relation) r_i of R_i is a set of tuples such that for each attribute $A_k \in attr(R_i)$, $t[A_k] \in dom(A_k)$, $\forall\, t \in r_i$, where $t[A_k]$ denotes the projection of t onto A_k. A database instance r of $\mathcal{R}$ is a collection of relations $(r_1,\ldots,r_n)$, where r_i is a relation instance of R_i, for $i \in [1, n]$.

Aiming to improve the quality of database schemas and to reduce manipulation anomalies, in the context of relational databases, functional dependencies (FDs) have been used as means to guide data normalization processes. In particular, an FD over database schema $\mathcal{R}$ is a statement $X \rightarrow Y$ (X implies Y) defined between two sets of attributes $X, Y \subseteq attr(\mathcal{R})$, such that, given an instance r of $\mathcal{R}$, $X \rightarrow Y$ is satisfied in r if and only if for every pair of tuples (t_1, t_2) in r, whenever $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$. X and Y represent the Left-Hand-Side (LHS) and Right-Hand-Side (RHS) of the FD, respectively.

Starting from the *canonical* definition of FD over 35 extended definitions have been provided in the literature, which have been generalized under the concept of *relaxed functional dependency* (RFD) [7]. In particular, RFDs enable the consideration of some relaxation criteria, which can lead to i) the consideration of similarity/difference constraints as attribute comparison method, and/or ii) the possibility that the dependency might hold for a subset rather than all the tuples.

RFD *definition.* Consider a relational database schema $\mathcal{R}$, and a relation schema $R = (A_1, \ldots, A_m)$ of $\mathcal{R}$. An RFD φ applied on $\mathbb{D}_c \subseteq dom(R)$ is denoted by

$$X_{\Phi_1} \xrightarrow{\Psi \leq \varepsilon} Y_{\Phi_2} \tag{1}$$

where

- $\mathbb{D}_c = \left\{ t \in dom(R) \mid \bigwedge_{i=1}^{m} c_i(t[A_i]) \right\}$ with $c = (c_1, \ldots, c_m)$, and each c_i is a predicate on $dom(A_i)$;
- $X = B_1, \ldots, B_h$ and $Y = C_1, \ldots, C_k$, with $X, Y \subseteq attr(R)$ and $X \cap Y = \emptyset$;
- $\Phi_1 = \bigwedge_{B_i \in X} \phi_i[B_i]$ ($\Phi_2 = \bigwedge_{C_j \in Y} \phi_j[C_j]$, resp.), where ϕ_i (ϕ_j, resp.) is a conjunction of predicates on C_i (C_j, resp.) with $i = 1, \ldots, h$ ($j = 1, \ldots, k$, resp.). For any pair of tuples $(t_1, t_2) \in dom(R)$, the constraint Φ_1 (Φ_2, resp.) is true if $t_1[B_i]$ and $t_2[B_i]$ ($t_1[C_j]$ and $t_2[C_j]$, resp.) satisfy the constraint ϕ_i (ϕ_j, resp.) $\forall$ $i \in [1, h]$ ($j \in [1, k]$, resp.).
- Ψ is a coverage measure defined on $dom(R)$, quantifying the amount of tuples violating or satisfying φ. It can be defined as a function $\Psi : dom(X) \times dom(Y) \rightarrow \mathbb{R}^+$, where $dom(X)$ is the cartesian product of the domains of attributes composing X.
- ε is a threshold indicating the upper bound (or lower bound in case the comparison operator is $\geq$) for the result of the coverage measure.

Given $r \subseteq \mathbb{D}_c$ a relation instance on R, r *satisfies* the RFD φ, denoted by $r \vDash \varphi$, if and only if: $\forall\, t_1, t_2 \in r$, if Φ_1 indicates true, then *almost always* Φ_2 indicates true. Here, *almost always* is expressed by the constraint $\Psi \leq \varepsilon$.

As an example, in a database of publications, it is likely to have a similar Affiliation for authors with the same Email address and similar Name. In particular, the functional determination should tolerate possible exceptions since authors might change affiliation over the years. This can be modeled by means of the following RFD:

$$\mathsf{Name}_{\leq 4}, \mathsf{Email}_{\leq 0} \xrightarrow{\psi(\text{Name, Email, Affiliation}) \leq 0.03} \mathsf{Affiliation}_{\leq 5}$$

where the comparison constraints (ϕ_1, ϕ_2, and ϕ_3) use the edit distance as function, the $\leq$ as comparison operator, and 4, 0, 5 as thresholds for Name, Email, and Affiliation, respectively. In general, the search space of the dependency discovery strategy can be modeled as a graph representation of a lattice, which is partitioned into levels where level L_i contains all attribute combinations of size i. Each node in the lattice represents a unique set of attributes, and it is linked to nodes that contain a direct superset or subset of attributes. In other words, each edge refers to the inclusion relation between two attribute sets. Thus, a lattice permits to consider candidate RFDs at each level in terms of lattice's edges, allowing to represent the LHS and the RHS of an RFD [19]. It is worth to notice that this representation is complete for FDs and RFDs relaxing on the extent. In fact, to discover RFDs relaxing on the attribute comparison it is necessary to also consider all possible dispositions of similarity/distance constraints among attributes included in a combination. Nevertheless, for this kind of RFDs we simplified the representation by visualizing the presence of an RFD as an edge in the lattice when there exists at least one valid RFD involving the same attribute combinations.

More formally, let $R = A_1, \ldots, A_m$ be a relation schema with m attributes. The corresponding graph representation of lattice will contain a collection of attribute sets, where *Level 0* contains the empty set, *Level 1* singleton sets, one for each attribute, *Level 2* the pair sets, one for each possible combination of two attributes, and so forth. Finally, the last level, namely *Level M*, will contain a single set of all the attributes from R. A lattice edge links two attribute sets on two adjacent lattice levels. For instance, let AB and ABC be attribute sets on *Level 2* and *Level 3*, respectively, then the edge $e(AB, ABC)$ represents the candidate $AB \rightarrow C$.

The number of RFDs holding on a given set of data can be very huge, especially when relaxation criteria settings increase. Thus, aiming to provide compact representation on where the holding RFDs converge in the search space, we propose DEVICE, which is described in the next section.

4. A tool for analyzing RFDs during discovery processes

In this section, we describe DEVICE. It enables monitoring of RFDs validated during the execution of a discovery algorithm. In particular, we first present the system architecture (Section 4.1), and then provide details on the visual interface (Section 4.2), and the interactions that a user can perform (Section 4.3).

4.1. System architecture

Monitoring the RFD discovery results during the execution of discovery algorithms is a complex problem. In fact, it is necessary to deal with several issues that affect the choices of the system architecture: i) the existing discovery algorithms are based on different technologies and frameworks, so requiring the integration of at least one module to adapt the system to the different algorithms; ii) the presence of multiple visualization components expects frequent updates in a short time, and iii) the number of dependencies processed can be large at any instant of time. For these reasons, we proposed a modular client-server architecture for enabling users to monitor discovery algorithms during their executions, and interact with the results through a responsive visual interface. The architecture of the DEVICE is shown in Figure 1. In detail, the architecture is composed of several standalone modules, which share information with other ones by exploiting the JSON standard. This type of solution allows DEVICE to ensure high component modularity and

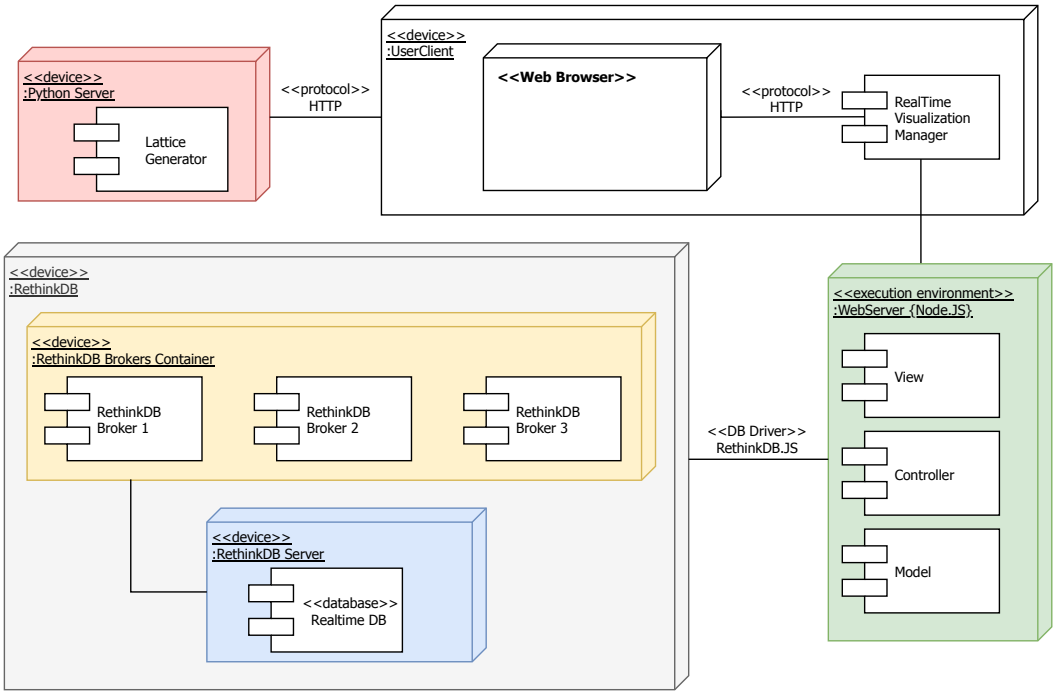


Figure 1: The system architecture of DEVICE.

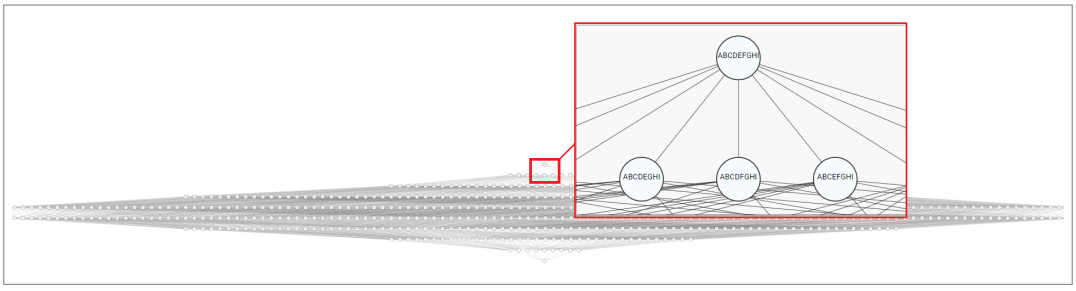


Figure 2: An example of lattice visualization.

maintainability, with the aim of updating or replacing any back-end module by simply adapting its output according to the JSON standard defined for the interaction.

The interface module on the client-side communicates with two different back-end subsystems. The first one allows DEVICE to automatically generate a lattice representation in JSON format by only considering the number of attributes. The set of nodes contains all the possible combinations of attributes on the lattice, while the set of edges contains all the existing links between two nodes of successive levels. The *Lattice Generator Server* receives a request containing the number of attributes for the lattice, creates the JSON, and returns its representation to DEVICE.

The second subsystem allows DEVICE to communicate with the discovery algorithms by exploiting a set of distributed message brokers. In particular, DEVICE is a web application distributed on multiple Node.JS server instances, which exploits the scalability of this technology combined with the speed of the RethinkDB[1] real-time database, to create a low latency and high-performance application. Although the architecture ensures flexibility, to make DEVICE compatible with most FD and RFD discovery algorithms, it has been necessary to integrate several communication modules to adapt the syntax of the dependencies of each algorithm, and to continuously monitor the results of each execution. To this end, the *Input Driver Connector* receives the dependencies from the algorithm, manipulates their syntax, and extracts a JSON version so as to store it in RethinkDB. The latter provides an internal set of message brokers that continuously store and send messages to the instance of Node.JS servers. The *Real-Time Visualization Manager* listens for messages from brokers, and decides which visual component manipulates in the interface.

As said before, the proposed tool is also able to handle continuous discovery algorithms [16] and therefore it requires to maximize fluidity and to minimize processing times

[1] https://rethinkdb.com

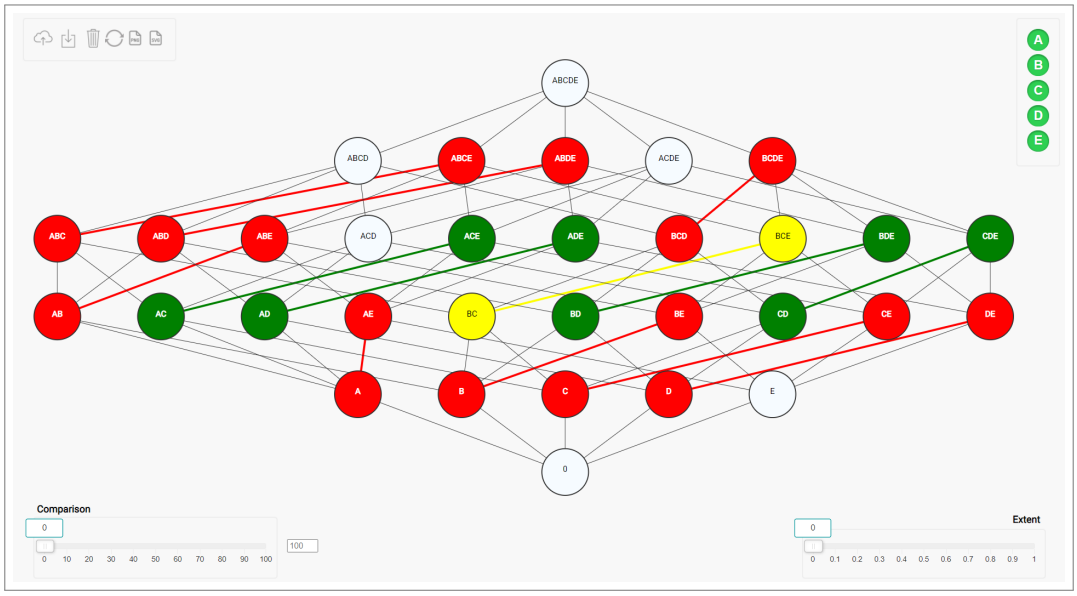


Figure 3: The visual interface of DEVICE during the execution of a discovery process.

within the visual interface. Thus, all selected technologies for both client- and server-side support real-time updating of data.

4.2. RFD Visualization

Due to the possible huge number of holding RFDs on a given set of processed data, systems for RFD visualization should enable users to analyze results in a compact way, by also giving the possibility to interact with them. For this reason, a static representation of discovery results after the execution of algorithms limits the analysis of how dependencies evolve over time, which is particularly interesting in dynamic contexts, like in the case of data streams.

The dynamic representation of a large portion of data requires the application of interactive graphs, capable of highlighting how information change over time. Hence, a dynamic visual representation of the search space has been implemented through a lattice graph representation. It enables a compact visualization on how holding RFDs converge into the search space (see Figure 8). As said before, the lattice permits to show candidate RFDs through the lattice edges, which connect attribute combinations differing by an attribute, so to represent in a compact way the LHS (common attributes) and the RHS (different attribute) of a candidate RFD. The lattice graph is responsible for displaying information about the candidate RFDs that have been validated during a discovery process. As shown in Figure 3, the lattice graph can show different colors during the execution of a discovery process. In particular, an edge is green when the corresponding candidate RFD has been evaluated and validated by the discovery algorithm. Instead, it assumes a red color when the RFD has been evaluated, but it is not valid. Finally, yellow edges represent candidate RFDs that are being analyzed. Aiming to emphasize the current validation results, DEVICE also uses colors for lattice nodes. In fact, a node assumes the same color as the last analyzed candidate RFD involving it.

It is worth to notice that, although we expect that different algorithms produce the same resulting set of discovered RFDs, when they analyze the same data, it is not obvious how they move in the search space. Thus, DEVICE enables the comparison among different discovery algorithms and the analysis of possible bottlenecks during their execution on a given set of data.

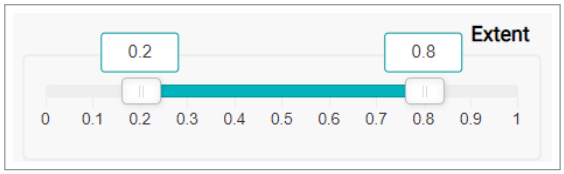
The visual interface of DEVICE also provides different gadgets enabling users to interact with the lattice graph. Moreover, the vectorial representation of the graph also permits to zoom on or move each lattice component without losing the quality of the representation. Details on how users can interact with the lattice graph are provided in the following.

4.3. Interaction in depth

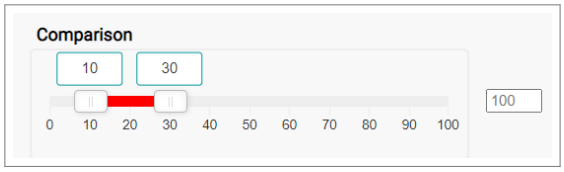
As mentioned above, aiming to emphasize the discovery results to a specific part of the search space, users can interact with the lattice graph by simply zooming on a specific part of the search space, or by moving its components into the visual interface. To this end, the user places the mouse pointer in correspondence with a lattice node and drags it in another place. Consequently, also edges linked to it are deformed by following the movement. Moreover, it is possible to filter out some nodes, so reducing the representation of the search space, by using the button list on the top-right of the visual interface (also shown in Figure 4(a)). In particular, each button represents an attribute of the analyzed data, and the user can select/deselect each of them to be included/excluded during the monitoring process. By default, all attributes appear in the search space. As an example, Figure 5 shows how the lattice is changed after the exclusion of the node A.

Concerning the RFD settings, DEVICE permits to visualize discovered RFDs by filtering results according to spe-

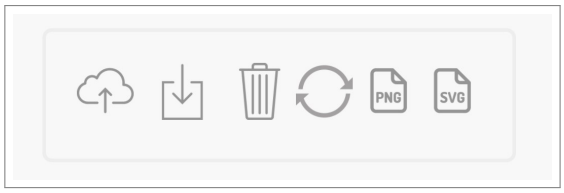
(a) Button list enabling the filtering of nodes in the lattice.



(b) Slider to define the range for the extent threshold.



(c) Slider to define the range the the comparison thresholds.



(d) Storing and execution icons.

Figure 4: DEVICE gadgets to interact with the lattice graph.

cific relaxation parameters. Figure 4(b) highlights a slider that enables the user in the definition of a specific range for the coverage measure threshold. In this way, lattice components' colors appear in accordance with the validation of RFDs having a satisfiability degree that meets the specified range bounds. Similarly, it is possible to filter out validation results in accordance with a range of thresholds composing difference constraints for the relaxation of the attribute comparison method (see Figure 4(c)). In particular, the bounds defined through the slider represent the range of possible thresholds that must appear on each attribute involved in candidate RFDs. However, as mentioned above, for sake of simplicity, a lattice edge is colored when at least one candidate RFD satisfies difference thresholds bounded by the range. Sliders are particularly useful for the analysis of RFD discovery results. In fact, with simple interactions, the user can evaluate how the set of holding RFDs can change as the relaxation settings are modified. Moreover, it is worth to notice that the interaction with sliders is enabled on the basis of the monitored algorithm. For instance, when a FD discovery algorithm is monitored, then sliders are set to $[0, 0]$ and cannot be modified. In this way, no errors and an exact comparison method (difference equal to 0) are admitted as RFD relaxation settings to represent FDs. In general, the same ranges are also used by default on both sliders, and it is possible to interact with them in accordance with the RFD category a discovery algorithm is devoted.

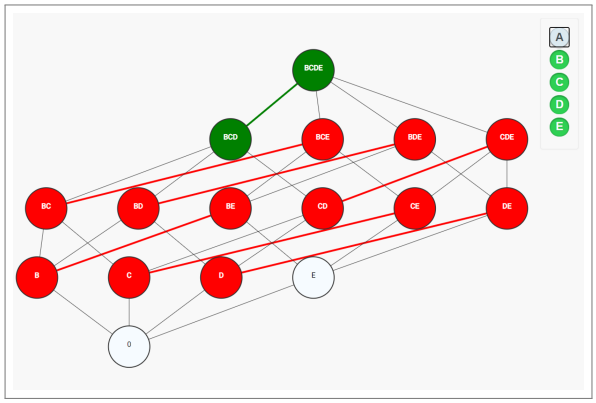


Figure 5: The visual interface of DEVICE after filtering out the attribute A.

Finally, the icons in Figure 4(d) enable the interaction with the monitored execution and the downloading of the lattice graph in several formats. In particular, the first two icons permit to upload or download the discovery results in a JSON file, respectively. The third and fourth icons enable the user to interact with the monitoring process. More specifically, the third icon permits to refresh the monitoring, by cleaning the lattice representation, and the fourth one gives the possibility to reload the lattice representation of the last execution of a discovery algorithm. Moreover, the colored lattice representation can be downloaded as an image in the *.png* or *.svg* format by using the second-last and the last icon, respectively.

5. Monitoring discovery algorithms

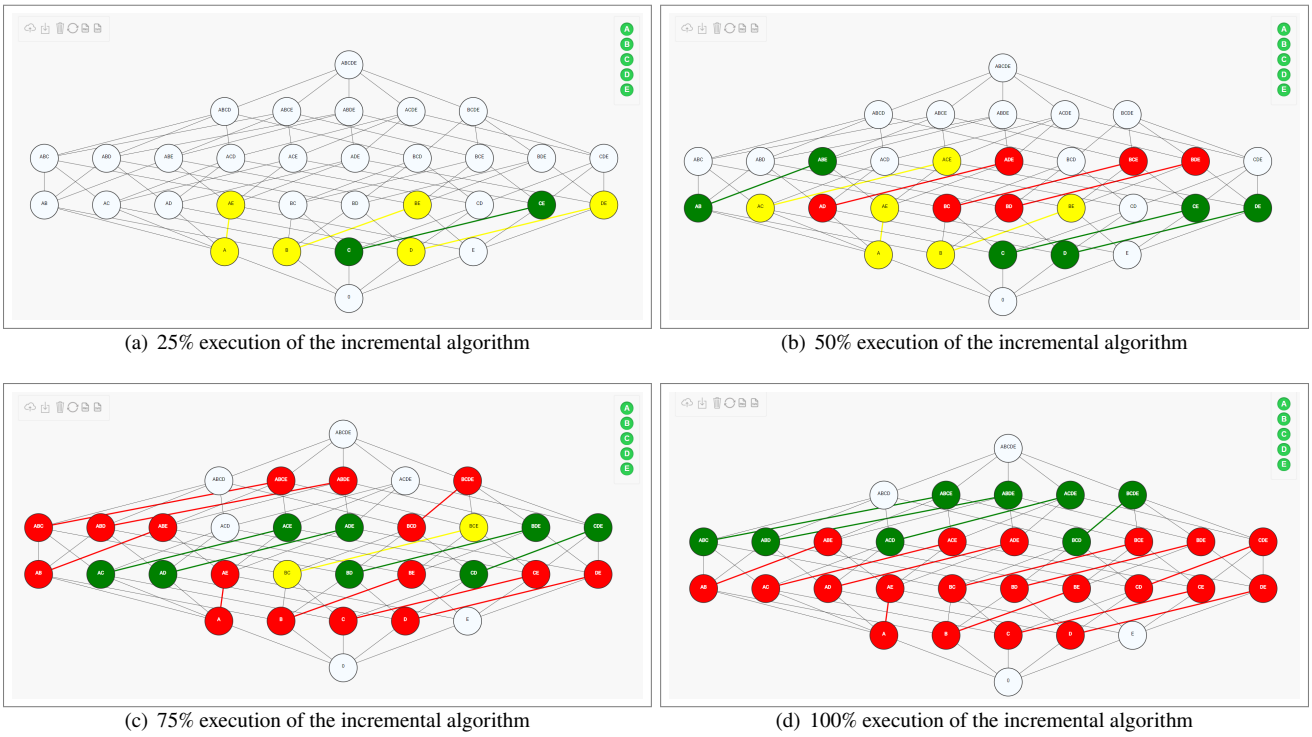
In this section, we show the effectiveness of DEVICE on different algorithms, by considering two different case studies on real-world datasets and on real sensor-based streams, with the aim of analyzing how metadata evolves over time. Moreover, a demonstration video of DEVICE[2] allows us to show how the users can interact with the tool during monitoring processes.

5.1. Case study on a real-world dataset

In order to verify the effectiveness of DEVICE on a real scenario, we analyzed discovery results on a real-world dataset. We selected two different discovery algorithms to analyze how their discovery strategy browses the search space, aiming to extract the holding RFDs.

The first algorithm involved in our evaluation is the genetic algorithm proposed in [5]. This type of algorithm has the potential to effectively tackle the problems arising in RFDs discovery, since they are particularly efficient for global searches in large search spaces by exploiting operations inspired to natural species evolutions, such as natural selection, crossover, and mutation. The discovery process starts with a population of RFD candidates which is randomly generated during the initialization phase and evolves by stochastically selecting

[2] https://youtu.be/QC2FjF50A60



(a) 25% execution of the incremental algorithm

(b) 50% execution of the incremental algorithm

(c) 75% execution of the incremental algorithm

(d) 100% execution of the incremental algorithm

Figure 6: Monitoring the incremental discovery algorithm during its executions.

multiple candidates from the current population. The algorithm exploits a fitness function to quickly validate each RFD and select the ones to involve during the evolution phases.

The second algorithm exploits incremental discovery strategies to extract functional dependencies from static and/or dynamic datasets [3]. Unlike the genetic algorithm, the incremental approach takes in input a set of candidates valid at a given instant of time, and returns the dependencies valid after updating at least one tuple. However, during the initial execution, the algorithm starts by considering the set of FDs candidates at the lower level of lattice, i.e. all the FDs with a single attribute on the LHS, and performs an upward search strategy of the lattice.

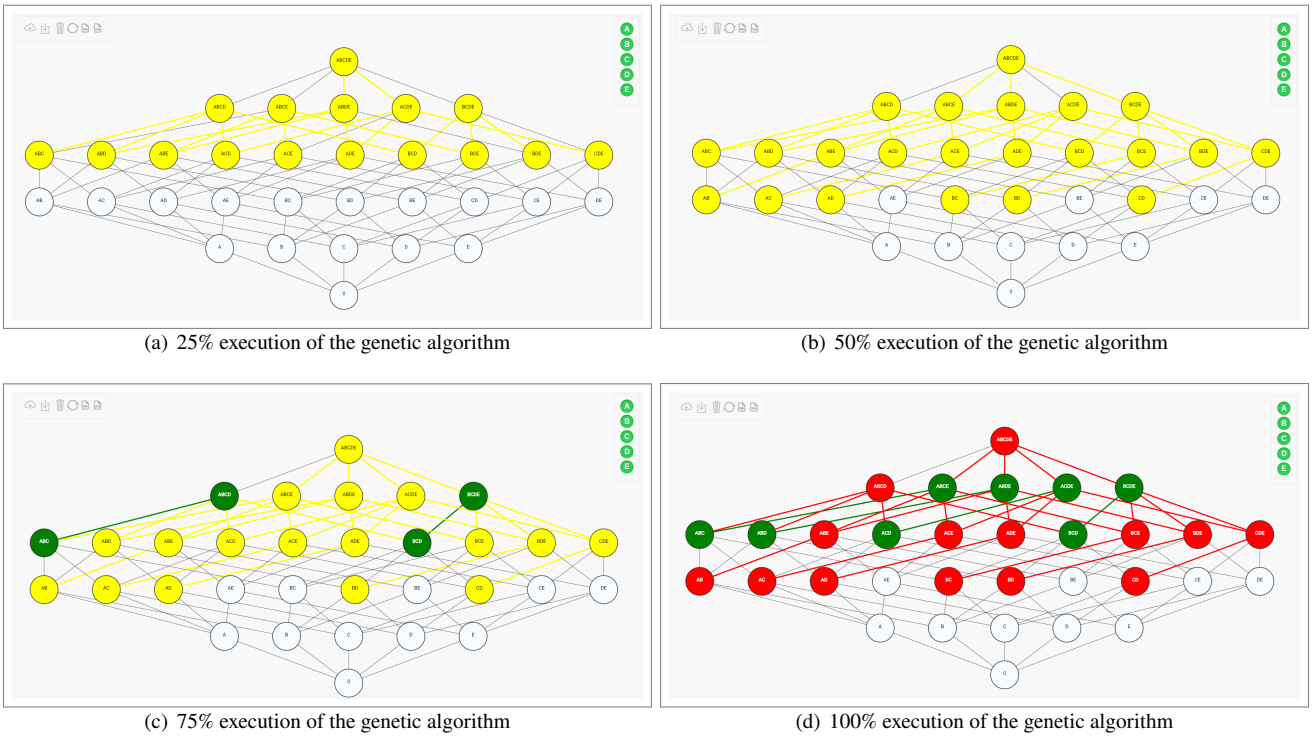
According to the characteristics of the considered algorithms, and to make processes comparable, we set parameters of genetic algorithm to discover canonical FDs. In particular, we choose the above-mentioned types of algorithms since they both use several iterations to get results. Nevertheless, their nature is quite different since the genetic algorithm analyzes new RFD candidates at each iteration by always considering the complete set of tuples; instead, the incremental algorithm analyzes new tuples at each iteration by considering the RFDs holding at the previous iteration (time-instant). Our aim is to show the usefulness of DEVICE in helping users to get insights on how algorithms can explore the search space.

Although these algorithms have been created with two different technologies, the *Input Driver Connector* allowed us to quickly adapt their output modules to DEVICE. In fact, this enabled us to monitor their executions on the same dataset, and compare how they browse the search space. To perform our evaluation, we ran each algorithm on the *Iris* dataset by automatically storing the screen of the lattice approximately every 1 second. Each screen represents the status of the lattice at any instant of execution time. For the sake of clarity, we only report the screens at 25%, 50%, 75%, and 100% of their executions (Figures 6 and 7).

More specifically, figures 6(a) and 7(a) show the evolution of the discovery process for the incremental and genetic algorithms, respectively, at the 25% of their executions. We can notice the difference between the two discovery strategies. In fact, the genetic algorithm starts to consider candidates in the middle of the lattice, and next perform a random upward search. However, the incremental algorithm first selects candidates from the lowest level, then goes up by performing a targeted search.

Another relevant difference between the two search strategies concerns the validation strategy of the candidates. In fact, the genetic algorithm exploits an a posteriori validation strategy of the RFD candidates, which allows to define the first valid and invalid dependencies only after 50% of the execution (Figures 7(b), 7(c), and 7(d)). On the contrary, the incremental algorithm updates the RFDs validated at the earliest executions according to the dynamic change of the dataset, and exploits this information to move on the search space (Figures 6(b), 6(c), and 6(d)). However, as expected when algorithms end the executions (Figures 6(d) and 7(d)) they obtain the same set of resulting RFDs.

DEVICE provides a concrete representation of the discovery algorithms, allowing users and domain experts to easily



(a) 25% execution of the genetic algorithm

(b) 50% execution of the genetic algorithm

(c) 75% execution of the genetic algorithm

(d) 100% execution of the genetic algorithm

Figure 7: Monitoring the genetic discovery algorithm during its executions.

monitor each execution phase, and to concretely compare the different search strategies.

5.2. Case study on a real-world data stream

In our last experiment, we show the usefulness of DEVICE on a real-world data stream. In particular, we executed the algorithm in [3] on data from 1,000 real sensors spread throughout Italy, made available by the *Openweathermap* portal[3]. These types of sensors share information about the weather forecast during the day. The data are frequently updated based on global and local weather models, satellites, radars, and a vast network of weather stations. In particular, we selected the following 8 attributes from the data stream:

- Temperature represents the temperature value in the Kelvin scale (K);
- Feels_like represents the human perception of weather in Kelvin scale (K);
- Sea_level represents the atmospheric pressure on the sea level (hPa);
- Grnd_level represents the atmospheric pressure on the ground level (hPa);
- Humidity represents the rate of humidity;
- Date represents the date of the weather forecast;
- Weather represents the weather condition (e.g. Rain, Snow, Extreme, etc.);
- Clouds_percentage represents the rate of cloud cover.

We considered a single execution of the algorithm on weather data streams lasting 4 days. The execution involved over 40,000 tuples shared by over 1,000 sensors. During the test, DEVICE continuously monitored the progress of the discovery algorithm, also storing the results and its status for different time intervals. Figure 8 shows the resulting FDs for each time interval. We can notice that the number of resulting FDs has a negative trend since the continuous insertion of new tuples has lead to many invalidations. Moreover, the algorithm in [3] incrementally discovers FDs, which require to be validated on the entire stream. This means that the initial number of FDs, i.e. dependencies involving few attributes, will probably evolve as the algorithm considers new tuples. To get some insights on the FD validation trend, DEVICE allows us to interact with its interface and explore the search space to concretely analyze how FDs evolved in this process.

Figure 9 shows the details of the discovery process by considering three different time intervals, 3, 48, and 96 hours, respectively. As expected, DEVICE shows that the algorithm has a large variation in the number of FDs after 3 hours and a small number of invalid FDs (Figure 9(a)). Moreover, as we can see, a relevant part of the search space has not been analyzed. This is due to the fact that the discovery strategy has already validated some minimal FDs, avoiding the analysis of candidates that can be directly inferred. Figure 9(b) and 9(c) show that many of the FDs validated after 3 hours, have been invalidated. Moreover, Figure 9(c) shows that the algorithm also analyzed many of the candidate FDs in the search space not analyzed before. This is due to the invalidation of

[3] https://openweathermap.org/

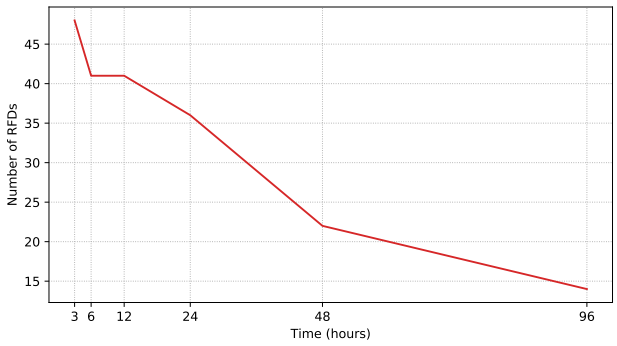


Figure 8: Resulting RFDs from the executions on real streams.

many dependencies on the right side of the search space. In fact, after 96 hours, only 14 FDs have been validated.

The evaluation performed on these real-world streams permits to understand how this kind of tool is able to support users and domain experts in the analysis of correlations holding on data streams. In fact, at each instant, an expert can concretely visualize and evaluate discovery results, and s/he can also monitor the evolution of holding RFDs over time. Moreover, the different gadgets embedded in DEVICE support users to interact with results during the monitoring process. For instance, the zoom feature (see Figure 9(d)) permits to focus the monitoring only on a specific part of the search space; instead, the filter feature (see Figure 9(e)) permits to isolate a specific set of RFD candidates. These two features enable to perform detailed analysis in order to consider the possibility to re-execute discovery processes on the same stream configurations, but with a reduced set of attributes. In general, these kinds of interactions could allow users to reduce the complexity of the analysis especially when they have to monitor big datasets and/or data streams.

6. Conclusion and Future Directions

Information visualization techniques aim, among others, to facilitate analytical processes and to reduce their interpretation complexity by exploiting, possibly specific or novel, visual representations. Nevertheless, the current big data contexts entail several challenging scenarios, where data are dynamically produced. In particular, in the context of dependency discovery from data streams (i.e., continuous profiling), dynamic data might produce the evolution of many FDs and/or RFDs. Thus, it is necessary not only to adequate discovery algorithms to fast execution processes, but also to allow users to analyze holding RFDs, and how they change over time. To this end, we have proposed the tool DEVICE, which relies on a lattice graph representation of the search space to let users actively visualize holding RFDs in a compact way during the execution of (incremental) RFD discovery algorithms. DEVICE also represents a useful means to compare different discovery algorithms, and to analyze how they browse the search space.

In the future, we would like to test the usability of DEVICE, by involving domain experts and scientists in the interpretation of discovery results on both incremental scenarios and algorithm comparison tasks. Thus, based on these results, we would like to extend DEVICE to better support its usefulness in the analysis tasks. Moreover, we would like to lighten the representation of the search space, when it has to represent big datasets. To this end, we are working on different grouping functionalities, which would enable the lattice graph with the possibility to dynamically change its shape according to the RFDs validated over time.

References

[1] Breve, B., Caruccio, L., Cirillo, S., Deufemia, V., Polese, G., 2020. Visualizing dependencies during incremental discovery processes., in: Proceedings of the Workshops of the EDBT/ICDT 2020 Joint Conference.

[2] Caruccio, L., Cirillo, S., 2020. Incremental discovery of imprecise functional dependencies. Journal of Data and Information Quality (JDIQ) 12, 19:1–19:25.

[3] Caruccio, L., Cirillo, S., Deufemia, V., Polese, G., 2019a. Incremental discovery of functional dependencies with a bit-vector algorithm, in: Proceedings of the 27th Italian Symposium on Advanced Database Systems.

[4] Caruccio, L., Deufemia, V., Polese, G., 2016. On the discovery of relaxed functional dependencies, in: Proceedings of the 20th International Database Engineering & Applications Symposium, IDEAS 2016, Montreal, QC, Canada, July 11-13, 2016, ACM. pp. 53–61.

[5] Caruccio, L., Deufemia, V., Polese, G., 2017. Evolutionary mining of relaxed dependencies from big data collections, in: Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, pp. 1–10.

[6] Caruccio, L., Deufemia, V., Polese, G., 2019b. Visualization of (multimedia) dependencies from big data. Multimedia Tools and Applications 78, 33151–33167.

[7] Caruccio, L., Deufemia, V., Polese, G., 2020. Mining relaxed functional dependencies from data. Data Mining and Knowledge Discovery 34, 443–477.

[8] Chakraborty, S., Tomsett, R., Raghavendra, R., Harborne, D., Alzantot, M., Cerutti, F., Srivastava, M., Preece, A., Julier, S., Rao, R.M., et al., 2017. Interpretability of deep learning models: a survey of results, in: 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), IEEE. pp. 1–6.

[9] Chen, W., Xie, C., Shang, P., Peng, Q., 2017. Visual analysis of user-driven association rule mining. Journal of Visual Languages and Computing 42, 76–85.

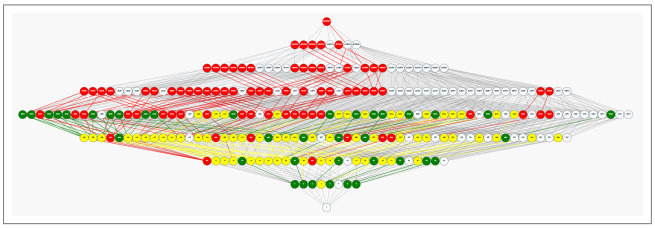
[10] Cirillo, S., Desiato, D., Breve, B., 2019. CHRAVAT – chronology awareness visual analytic tool, in: 2019 23rd International Conference Information Visualisation (IV), IEEE. pp. 255–260.

[11] Costagliola, G., Fuccella, V., Giordano, M., Polese, G., 2008. Monitoring online tests through data visualization. IEEE Transactions on Knowledge and Data Engineering 21, 773–784.

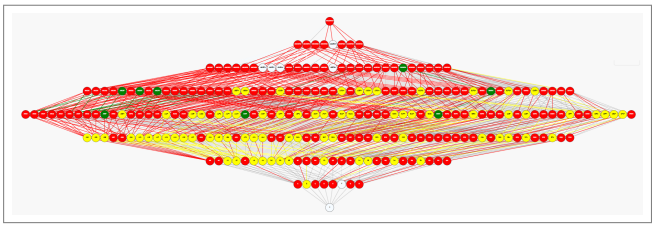
[12] De Oliveira, M.F., Levkowitz, H., 2003. From visual data exploration to visual data mining: a survey. IEEE Transactions on Visualization and Computer Graphics 9, 378–394.

[13] Di Rocco, L., Dassereto, F., Bertolotto, M., Buscaldi, D., Catania, B., Guerrini, G., 2020. Sherloc: a knowledge-driven algorithm for geolocating microblog messages at sub-city level. International Journal of Geographical Information Science , 1–32.

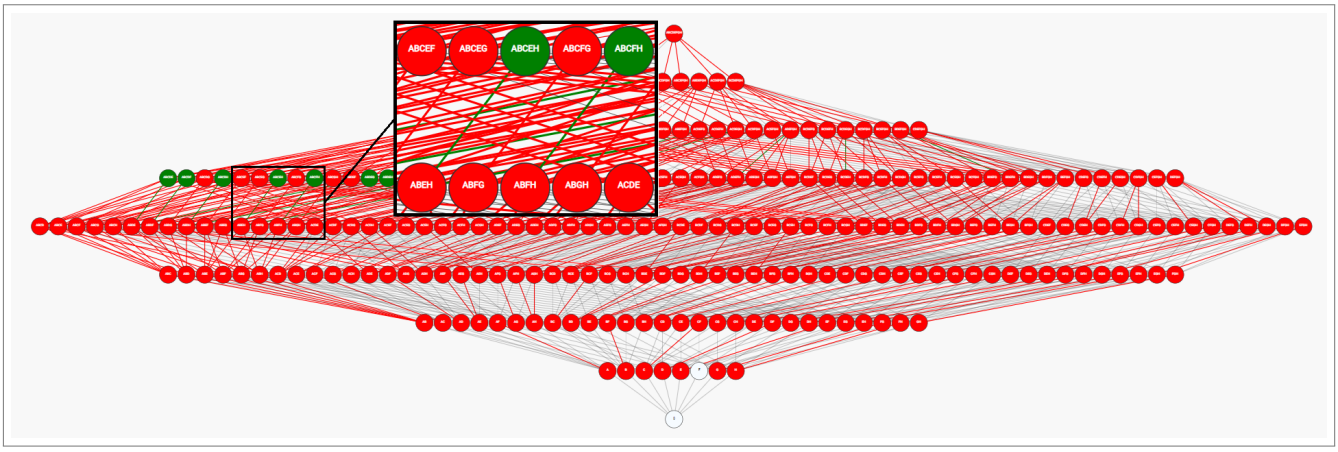
[14] Kruse, S., Hahn, D., Walter, M., Naumann, F., 2017. Metacrate: Organize and analyze millions of data profiles, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, ACM. pp. 2483–2486.



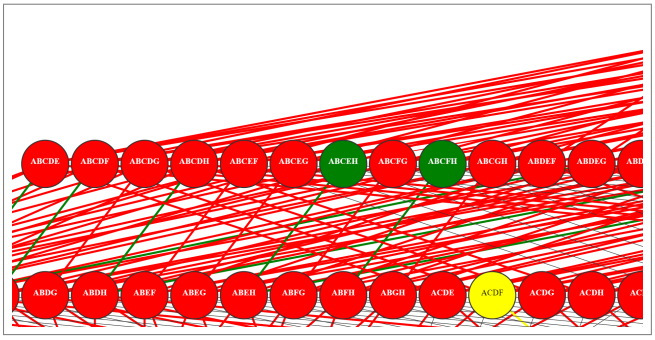
(a) Monitoring interface after 3 hours.



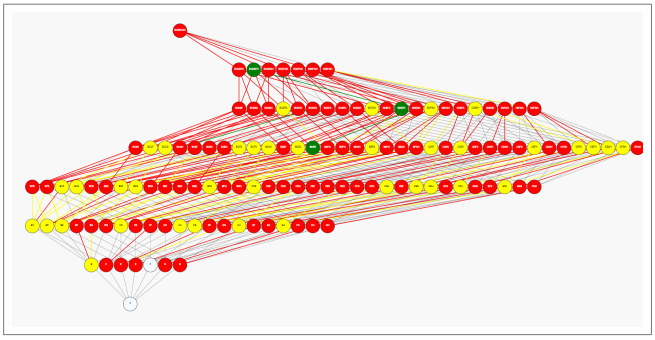
(b) Monitoring interface after 48 hours.



(c) Monitoring interface after 96 hours.



(d) Monitoring interface after zooming.



(e) Monitoring interface after filtering.

Figure 9: Monitoring the incremental discovery algorithm during its executions on real streams.

[15] Kruse, S., Naumann, F., 2018. Efficient discovery of approximate dependencies. Proceedings of the VLDB Endowment 11, 759–772.

[16] Naumann, F., 2014. Data profiling revisited. ACM SIGMOD Record 42, 40–49.

[17] Nicolazzo, S., Nocera, A., Ursino, D., Virgili, L., 2020. A privacy-preserving approach to prevent feature disclosure in an IoT scenario. Future Generation Computer Systems 105, 502–519.

[18] Papenbrock, T., Bergmann, T., Finke, M., Zwiener, J., Naumann, F., 2015a. Data profiling with metanome. Proceedings of the VLDB Endowment 8, 1860–1863.

[19] Papenbrock, T., Ehrlich, J., Marten, J., Neubert, T., Rudolph, J.P., Schönberg, M., Zwiener, J., Naumann, F., 2015b. Functional dependency discovery: An experimental evaluation of seven algorithms. Proceedings of the VLDB Endowment 8, 1082–1093.

[20] Papenbrock, T., Naumann, F., 2016. A hybrid approach to functional dependency discovery, in: Proceedings of the 2016 International Conference on Management of Data, ACM. pp. 821–833.

[21] Raghav, R., Pothula, S., Vengattaraman, T., Ponnurangam, D., 2016. A survey of data visualization tools for analyzing large volume of data in big data platform, in: Proceedings of the 2016 International Conference on Communication and Electronics Systems (ICCES), IEEE. pp. 1–6.

[22] Saxena, H., Golab, L., Ilyas, I.F., 2019. Distributed discovery of functional dependencies, in: IEEE 35th International Conference on Data Engineering, IEEE. pp. 1590–1593.

[23] Schirmer, P., Papenbrock, T., Kruse, S., Hempfing, D., Meyer, T., Neuschäfer-Rube, D., Naumann, F., 2019. DynFD: Functional dependency discovery in dynamic datasets, in: Proceedings of the 22nd International Conference on Extending Database Technology (EDBT '19), pp. 253–264.

[24] Sekhavat, Y.A., Hoeber, O., 2013. Visualizing association rules using linked matrix, graph, and detail views. International Journal of Intelligence Science 3, 34–49.

[25] Song, S., Chen, L., 2013. Efficient discovery of similarity constraints for matching dependencies. Data & Knowledge Engineering 87, 146–166.

[26] Wei, Z., Link, S., 2019. Discovery and ranking of functional dependencies, in: IEEE 35th International Conference on Data Engineering, IEEE. pp. 1526–1537.

Journal of Visual Language and Computing

journal homepage:

Auto-Modularity Enforcement Framework Using Micro-service Architecture

Hanzhong Zheng, Justin Kramer and Shi-Kuo Chang*

Department of Computer Science, University of Pittsburgh, 6135 Sennott Square, 210 S Bouquet St., Pittsburgh, PA, USA, 15260-9161

ARTICLE INFO

Article History:
Submitted 10.22.2020
Revised 11.15.2020
Second Revision 12.2.2020
Accepted 12.10.2020

Keywords:
Micro-service
Automatic software development
Service-oriented architecture
Modularity enforcement
Visual software development

ABSTRACT

The evolution of the software architecture has been progressively shifting to emphasize modularity, isolation, scalability, agility, and loose coupling. Service-oriented architecture (SOA) has started to gain popularity in this direction. Micro-services are a lightweight SOA that aim to largely scale applications while ensuring isolation and distribution. Modularity is sometimes left behind or difficult to achieve with fine-grained distribution of programmer responsibilities. In this paper, we propose an automatic modularity enforcement (AME) framework during the software development life cycle (SDLC) through intermediate representation. Our idea was inspired by automatic software development for building a scalable application. We implemented this framework to support visual software development using the Java Spring Boot Micro-service tool.

1. Introduction

Software architecture reflects the definition of all interacting components in the system for satisfying customers' requirements. Nowadays, analytic applications largely increase the criticality of the software quality and scalability. In the micro-service paradigm, scalability and isolation are improved through dividing a large application service into several sub-services, which are independently deployed and communicated through interfaces via standard data formats and protocols such as XML, HTTP, etc. [1]. Each sub-service implements the partial functionality of the entire system.

The majority of a software system is divided into several modules during the design. However, modularization has always been one of the greatest challenges in software architecture design. Enforcing modularization can also be considered as an NP-hard problem for programmers [2]. Modularization separates the program's functionality into several modules. Each module is independent and interacts with each other. Inadequate modularization can easily influence the distribution, persistence, isolation, and even the overall software quality. For programmers, modularization is a key but challenging principle. The complexity of modern software systems makes them much harder for programmers to understand and maintain, especially with respect to scalability. Modularization can allow for decomposition of the software system to reduce the complexity and improve the maintainability. Furthermore, modularity can make the application more tolerant of uncertainty.

To ensure the continuous delivery of trustworthy and high-quality software systems while reducing the burdens on programmers, automation in software development has become important. Many efforts have been made in recent years in automation of the software development process under three categories: Rapid Application Development (RAD) [3] [4], Code generation [5], and Model-Driven Architecture (MDA) [6] [7]. In object-oriented programming, modularity and encapsulation are closely tied to each other and play dominant roles. Enforcing modularity can limit the propagation of program errors and establish software maintainability. The mechanism of automatic module enforcement (AME) allows the program modules to be developed in a customized and organized way. AME sets more constraints in order to keep consistency and cohesion in the entire software system design. In this

*Corresponding author
Email address: schang@pitt.edu
ORCID: 0000-0003-0426-4030

DOI reference number: 10.18293/JVLC2020N2-008

paper, we propose an automatic module enforcement framework that automatically generates and enforces the modularity in software development from the software system design. This framework is flexible and agile for adapting into other programming languages. The contributions of this paper are as follows:

1. We developed a new time-critical application design system that specify different interaction patterns among components in the software system design.

2. We proposed an automatic modularity enforcement (AME) framework using the concept of micro-service architecture to generate and reinforce the module's functionality and cohesion.

3. We implemented our framework on a well-defined experimental system using the Java Spring Boot developing template.

2. Related Work

2.1 Service-Oriented Architecture (SOA)

Enterprises have increased their demand for flexible, efficient and extendable architecture paradigms in the current highly competitive software market. SOA is a service-based architectural style that usually is viewed as a black box that may have many underlying services [8], but brings many significant benefits to Enterprises in the way of flexibility, agility and high degree of collaboration between business and IT. The flexibility of SOA demonstrates how legacy applications can easily integrate with new applications. SOA has the ability to quickly respond to ever-changing requirements and demands. The main goal of SOA is to support a business process that reflects their collaboration.

The communications in a SOA commonly utilize WSDL (Web Service Description Language), UDDI (Universal Description Discovery and Integration), and SOAP (Simple Object Access Protocol) among Service

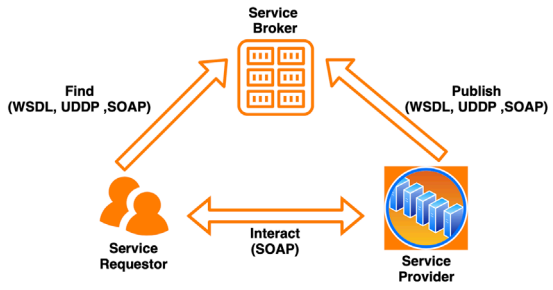


Figure 1: Communication structure of SOA

Provider, Service Broker and Service Requestor/Consumer (Fig. 1). Service Provider offers a variety of different services that are ready to use. Service Requestor demands the services. Service Broker is a service registry for connecting the Service Provider and Service Requestor.

The limitations of the SOA are also obvious. The communications between services mainly depend on message passing, which can easily become overwhelming when applications require heavy data exchange. The connections are exponentially increasing for a server due to transmission protocols, and SOA is costly in deployment and human resources.

2.2 Monolithic vs. Micro-service Architecture

The waterfall development model and associated technology are representations of traditional software development processes, which usually require a large team on a monolithic artifact. In the monolithic architecture, the main concept is "single": A monolithic application is built from a single unit, which is self-contained and independent from other applications. However, a great service design for a large application should be stateless and allow the application to scale vertically. Micro-services arrange an application to be a collection of loosely coupled, interconnected modular services where individual services communicate through REST APIs and lightweight messages. To achieve this isolation each service should be independent from other services. Fig. 2 illustrates an example of a micro-service architecture.

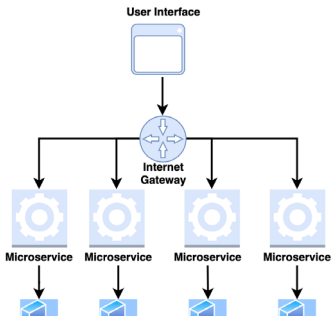


Figure 2: An example of a micro-service architecture

Maintainability, scalability and reliability are the main drawbacks of the monolithic architecture, and issues concerning them are proliferated in the current enterprise market. Micro-services ensure the continuous delivery and deployment of a large and complex application associated with scalability, testability, flexibility and fault tolerance. Service failure is unpredictable but harmful. Isolation in micro-services ensure the application continue to operate even if there is a service failure. Enterprise applications have the essence to be complex and highly demanding of

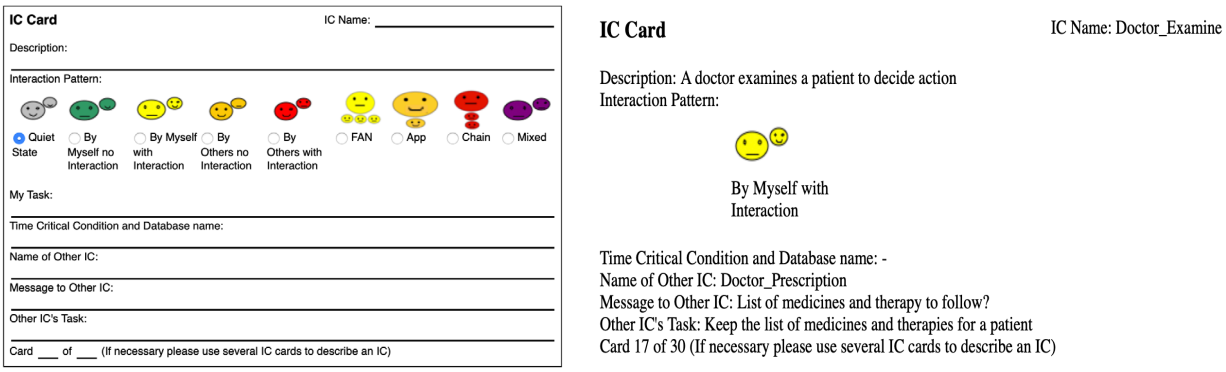


Figure 3: An IC card example of creating a functionality component for 'Doctor_Examine'

scalability and responsiveness. The benefits of micro-services seem to fulfill those requirements and attract business Enterprises transition from monolithic to micro-service architecture. As one of the biggest e-commerce company in Europe, Otto Group started to build their system using micro-services, which vertically decomposes their system into four loosely coupled applications: Product, Order, Promotion and Search/Navigation [9]. Another example is the Netflix. Netflix is the top Internet television network in the world and spent over 7 years on the transition to the micro-service. The successful transition allows the video to be displayed on a variety of screen sizes and platforms [10]. Besides, micro-services can easily integrate with popular cloud platforms. Amazon web services and Microsoft Azure both deploy micro-services on their cloud platforms.

2.3 Software Development Automation (SDA)

Software development in general requires large amounts of human endeavor. The improvement of computer architectures and networks drives the size of computer software and their diverse platforms [13] exponentially, increasing in order to satisfy the increasing needs of providing more software features. Automation in software development refers to replacing repeatable processes and reducing manual intervention, which accelerates the delivery of high-quality software products [11]. Automation in the software space focuses on building both software and testing automation, which usually takes a significant amount of time. Application building involves many steps, like code updating, compiling, and deployment. Software testing intends to discover bugs, errors, or defects during the execution of programs or application.

Software projects range from small scale personal projects to large scale industrial applications. This triggers the popularity of open software repositories such as Github, Sourceforge, and Bitbucket. However, well-maintained software development frameworks, like Sot, Wala, LLVM, all require a successful build process of the project repositories. Foyzul Hassan et al. present a feasible automatic software binding on Java projects on the state-of-the-art version control repository [12]. They found that 57% of build failures can automatically be resolved. Software testing is an intensive and costly task in the software development life cycle (SDLC). Automated software testing aims to reduce the workload through automated testing. Test automation largely impacts the quality, development time, and cost of the software to the market [16]. Automated software testing can be unit testing, functional testing, testing management tools, and so on.

Modularity is an important concept in software applications. It enhances the reusability of the previous code. Modules usually are divided based on their functionality, but they work together for serving a specified business domain. Modularization is always the main issue in SDLC and the core task for programmers [17] [18] [19]. One of the benefits of micro-services is the enhancement of modularity in the project to achieve fine-grained distribution of sub-services. For object-oriented programming, modularization is necessary for development teams. The high benefits of modularity certainly associate with the challenges in software design and implementation. We propose an automatic modularity enforcement framework from the software design to the software implementation process. The implementation of our framework utilizes micro-services to enforce isolation and reusability. We also demonstrate that flexibility by not only automatically creating Java modules, but also by automatically creating modules in other object-oriented programming languages. The IC card can model the interaction patterns for designers to choose, illustrated in different colors and emoticons with associated names. Interaction patterns define how statuses are communicated with other IC cards.

3. Time Critical Condition Design

Our IC card management system (ICMS) is used for designing time critical applications. An IC card is a visual specification scheme for rapidly prototyping the entities of an application [20]. The ICMS is a visual tool that allows the creation, edition, visualization, and exporting of one or more IC cards. The connection of

the ICMS and our auto-modularity enforcer utilizes XML specification. The ICMS automatically generates XML specifications. Fig. 3 is an example of the IC card example that can show the structure. There are 8 interaction patterns: 'quiet state', 'By myself no interaction', 'by Myself with interaction', 'By Others no interaction', 'By Others with interaction', 'FAN' (fan-out), 'App', 'Chain', and 'Mixed'. The 'Quiet' state indicates not working or in a restful state.

'FAN' indicates the distributed fan-out of a larger task to a number of smaller tasks. 'My task' is the task assigned to this IC card. The content of the IC card provides the detailed descriptions of the task. For example for a brainwave sensor component IC card, "(1) if T_c > threshold T; calculate two options states: attention and mediation states; otherwise, keep collecting the EEG. (2) if medication value > attention value, send the medication state and value to the database server; else: send the attention state and value to the database server." 'Name of Other IC' specifies the other IC cards that interact with each other. 'Messages to Other IC' contains the message format and message content (e.g. "msg1: raw EEG data, msg2: user state, value"). After finishing filling out the fields of the IC card, the designer submits the IC card to the database, and the IC card database automatically generates the other designated number of IC cards with temporary information so that the designer can edit them at any time. In addition, the XML schema has also been automatically generated and can be downloaded for the next component to transform into java modules.

4. Experimental Tool

For our experimental tool, we began with an analysis of the micro-service architecture. The micro-service paradigm focuses upon modularity in backend development by introducing componentization of the services it defines. These services are broadly defined within the framework, and the architecture behind these services varies vastly from one organization to another. With the growth of the micro-service architecture in recent years [9], the need for Rapid Application Development (RAD) [3] [4] in the space has expanded. Based upon this analysis, we developed a tool to enhance the reusability, cohesion, and distribution of micro-service development while reducing coupling. Our experimental tool is driven by an automatic modularity enforcement framework based upon the Java Spring Boot framework. Java Spring Boot supports the creation of a framework that utilizes fine-grained distribution of sub-services while allowing for vast extensibility through Cloud, API, and serverless interfaces to services.

We employed Python to engineer a dynamic auto-generation tool within our Java Spring Boot micro-service architecture. The Python program allows for flexible micro-service generation based upon a standard XML input interface, connecting to our IC card management system that produces the XML files. With our Python program, users can specify constant factors to identify their central repository, the XML file to target, and the title of their template files. To allow for extensibility, the Python Auto-Generator provides a basic method-layer API to process user-defined micro-services which fit into the templated IC card-based XML structure. This approach allows users to create micro-services through the framework of their choice if basic constraints on input and output are met.

Through the development of the experimental micro-service Auto-Generator (Fig. 4), it is possible to automatically generate micro-service components within a structure that lends itself to extensibility, tight cohesion, loose coupling, and modularity. The XML specification, based on the IC card template, serves as an interface to the Auto-Generator. An XML-based communication structure informs the Auto-Generator which micro-services to create, as well as the database tables to establish for each micro-service. Also, the Auto-Generator establishes micro-service file components in a directory structure specified in the source project directory.

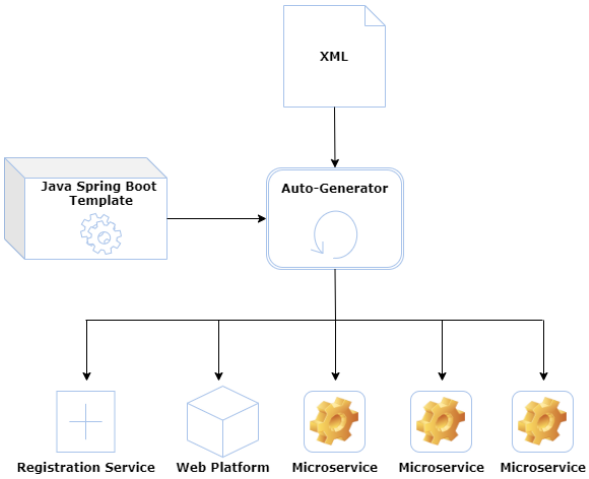


Figure 4: modularity auto-enforcement framework using micro-service architecture

As an example, the *fan* software structure is illustrated in Figure 5. This is for the *get user location* module of an app. At the top we have *track_tracks_service* and under that, as shown in Figure 6, we have two IC cards, *close_closes*, and *distance_distances*. The module (IC card) *track_tracks_service* has a database that stores the location, the latitude, and the longitude. The IC cards are shown in Figure 6. Once the IC cards are defined, the ICMS can output an XML specification as shown in

Figure 7.

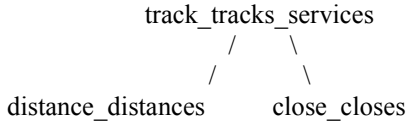


Figure 5. Fan software structure.

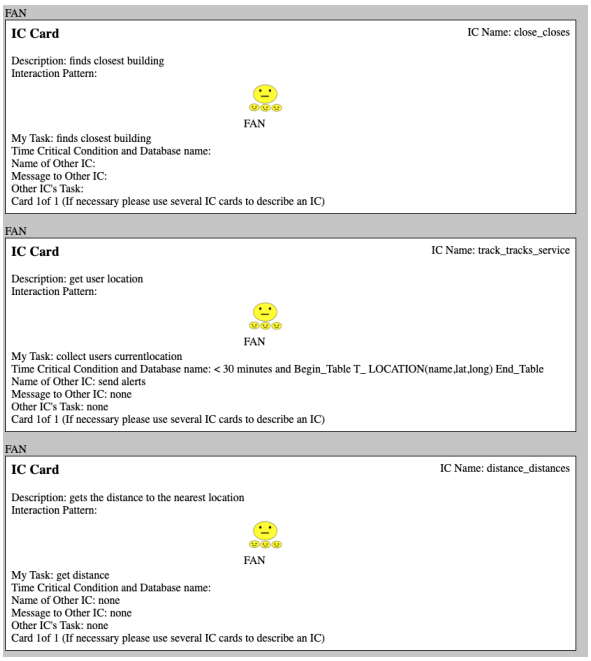


Figure 6. The IC cards for the *fan* software structure.

```
<?xml version="1.0" encoding="UTF-8"?>
<icCardList xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <icCardEntry icEntryId="2804" icEntryName="test">
      <icCard icId="10279" icName="close_closes"
icDescription="finds closest building" icIntPattern="FAN"
icMyTask="finds closest
       building" icTimeCriticalCondition="" icNumberCurrent="1"
icNumberTotal="1">
         <icOther icOtherName="" icOtherMessage=""
icOtherTask="" otherId="-1" />
      </icCard>
      <icCard icId="10277" icName="track_tracks_service"
icDescription="get user location" icIntPattern="FAN"
icMyTask="collect users
      currentlocation" icTimeCriticalCondition="&lt; 30 minutes and
Begin_Table T_ LOCATION(name,lat,longatiude) End_Table"
      icNumberCurrent="1" icNumberTotal="1">
         <icOther icOtherName="send alerts" icOtherMessage="none
" icOtherTask="none " otherId="4784" />
      </icCard>
      <icCard icId="10278" icName="distance_distances"
icDescription="gets the distance to the nearest location"
icIntPattern="FAN"
         icMyTask="get distance" icTimeCriticalCondition=""
icNumberCurrent="1" icNumberTotal="1">
         <icOther icOtherName="none" icOtherMessage="none"
icOtherTask="none" otherId="-1" />
      </icCard>
   </icCardEntry>
</icCardList>
```

Figure 7. The XML specfication.

Based upon the XML specification of the IC cards, the AutoGenerator can then create the output modules. A portion of a module is shown in Figure 8.

```
package io.pivotal.Micro-services.patients;

import java.io.Serializable;
import java.math.BigDecimal;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

/**
 * Persistent patient entity with JPA markup. Patients are stored in
an H2
 * relational database.
 *
 * @author Paul Chapman
 */
@Entity
@Table(name = "T_PATIENT")
public class Patient implements Serializable {

        public static Long nextId = 0L;

        @Id
        protected Long id;

        protected String number;

        protected String name;

        protected String address;

        /**
         * This is a very simple, and non-scalable solution to
generating unique
         * ids. Not recommended for a real application. Consider
using the
         * <tt>@GeneratedValue</tt> annotation and a sequence
to generate ids.
         *
         * @return The next available id.
         */
        protected static Long getNextId() {
                synchronized (nextId) {
                        return nextId++;
                }
        }

        /**
         * Default constructor for JPA only.
         */
        protected Patient() {
        }
```

Figure 8. A generated module.

The outputted modules, as demonstrated by the example module in Figure 8, provide two key functions. The first function is a set of class variables that define the database table for the service. The AutoGenerator establishes a table for the micro-service in the database for usage based on the template database. Furthermore, each module provides a method-level API that

controller classes utilize to manipulate and combine the data of each service. The core class of the module is extensible, allowing for the development of ICs within each core class. The ICs inside of each core class define the service's internal communication, logic, and functions. For example, an IC defined as RegisterUser may exist within the User Service. The User Service core class would contain the logic for registering a user, including CRUD (create, read, update, delete) calls to the User database. In addition, a Login Controller may act as a wrapper class around both the User Service and a hypothetical Authentication Service, which it utilizes to authenticate a potential user's information before registration. Lastly, the outputted module communicates with an automatically generated registration server to establish itself as a distributed micro-service. Connections to the registration server are based upon a centralized location that is automatically provided within the creation of each new micro-service.

5. Conclusion

This paper proposes the organization of the generated micro-service, with clear distinctions between Software Quality Assurance (SQA) testing, database creation, data seeding, registration, a web platform, and service methods. The combination of these resources is accessed by Java Spring Boot to compile the micro-service-based software. Based upon our experimental design, micro-services may be created in conjunction with the tenets of Rapid Application Development (RAD) [3] [4]. These micro-services register with a central server, utilize their independent databases, and provide APIs to controllers in the overlying software. Each micro-service interacts with the registration and web components through structured channels based upon naming schemas. Thus the experimental tool provides a basis for our studies pertaining to auto-modularity enforcement framework for micro-services.

Our next research goal is to investigate the optimal organization of the generated micro-services according to some objective functions to minimize, for instance, the total development efforts.

References

[1] Dragoni, N., Lanese, I., Larsen, S., Mazzara, M., Mustafin, R., Safina,L., 2017. Microservices: How to make your application scale.

[2] Prajapati, A., Chhabra, J., 2018. Optimizing software modularitywith minimum possible variations. Journal of Intelligent Systems 29.

[3] Beynon-Davies, P., Carne, C., Mackay, H., Tudhope, D., 1999. Rapidapplication development (rad): An empirical review. Eur. J. Inf. Syst. 8, 211–223.

[4] Berger, H., Beynon-Davies, P., Cleary, P., 2004. The utility of a rapidapplication development (RAD) approach for a large complex information systems development., 220–227.

[5] Liao, H., Jiang, J., Zhang, Y., 2010. A study of automatic code generation, in: 2010 International Conference on Computational and Information Sciences, 689–691. doi:10.1109/ICCIS/2010.171.

[6] Newman, M.E.J., Girvan, M., 2004. Finding and evaluating community structure in networks. Phys. Rev. E. 69, 026113.

[7] Pastor, O., Molina, J.C., 2007. Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer-Verlag, Berlin, Heidelberg.

[8] Cloutier, Robert. 2008. Model Driven Architecture for Systems Engineering. Presentation Slides), Stevens Institute of Technology, presented at INCOSE International Workshop.

[9] Chapter 1: Service Oriented Architecture (SOA). *msdn.microsoft.com*. Archived from the original on February 6, 2016. Retrieved September 21, 2016.

[10] Hasselbring, W., Steinacker, G., 2017. Microservice architectures forscalability, agility and reliability in e-commerce, in: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), 243–246.

[11] Vučković, J., 2020. You Are Not Netflix. Springer International Publishing, Cham. 333–346.

[12] Ohno, O., Furuhata, Y., Komuro, H., Imajo, T. and Komiya, S. 2002. Automated software development based on composition of categorized reusable components—construction and sufficiency of skeletons for batch programs. Electron. Comm. Jpn. Pt. II, 85: 50-66.

[13] Hassan, F., Mostafa, S., Lam, E.S.L., Wang, X., 2017. Automaticbuilding of java projects in software repositories: A study on feasibility and challenges, in: 2017 ACM/IEEE International Symposiumon Empirical Software Engineering and Measurement (ESEM), .38–47.

[14] Moran, K., 2018. Automating software development for mobile computing platforms (doctoral symposium). ArXiv abs/1807.07171

[15] Wedikian, Z., Ayari, K., Antoniol, G., 2009. Mc/dc automatic testinput data generation, in: Proceedings of the 11th Annual Conferenceon Genetic and Evolutionary Computation, Association for Comput-ing Machinery, New York, NY, USA. 1657–1664.

[16] Kumar, D., Mishra, K., 2016. The impacts of test automation on software's cost, quality and time to market. Procedia Computer Science79, 8–15.

[17] Garousi, V., Elberzhager, F., 2017. Test automation: Not just for testexecution. IEEE Software 34, 90–96. doi:10.1109/MS.2017.34

[18] Serme, G., 2013. Modularization of security software engineering indistributed systems. (modularisation de la sécurité informatique dansles systèmes distribués).

[19] Mitchell, B.S., Mancoridis, S., 2006. On the automatic modularization of software systems using the bunch tool. IEEE Trans.Softw. Eng. 32, 193–208.

[20] Hare, E., Kaplan, A., 2017. Designing modular software: A casestudy in introductory statistics. Journal of Computational and Graphical Statistics 26.

[21] Chang, S. K., Rajnovic, P., Zalar, M., 2007. IC Card: Visual specification for rapid prototyping of time-critical applications. International Journal of Software Engineering and Knowledge Engineering 17, 557–573.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc/

CACHE: Contextual Approach for Cultural Heritage Enhancing

Mario Casillo[a], Francesco Colace[a], Marco Lombardi[a,*] and Domenico Santaniello[a]

[a]DIIn - University of Salerno, Italy

ARTICLE INFO

Article History:
Submitted 10.26.2020
Revised 11.18.2020
Second Revision 12.5.2020
Accepted 12.10.2020

Keywords:
Geographic knowledge
Geographic rules
Formal grammar
Smart cities

ABSTRACT

In the panorama of Italian coastal tourism, there are many unique and unexplored places. These places, which suffer from the lack of government investment, present the need to be promoted through low consumption systems and widely used distributed applications.

The present work aims to develop innovative solutions to support citizens and tourists to offer advanced services, highly customizable, able to allow, through the use of new technologies, a more engaging, stimulating, and attractive use of information than the current forms. The developed system is based on graph-based formalisms such as Context Dimension Tree and Bayesian Networks, representing the context through its main components and react to it anticipating users' needs. Through the development of a mobile app, it was analyzed a case study applied in the area of Amalfi Coast (in Italy). Finally, an experimental campaign was conducted with promising results.

1. Introduction

Nowadays, Italian coasts represent one of the most characteristic and priceless tourist places globally, i.e., offering users an important cultural heritage. Unfortunately, not all sites receive the attention or financial support necessary to bring out their uniqueness. Thanks to the advent of new technologies and the smart cities phenomenon, these places could finally be protected and promoted. In fact, the adoption of Future Internet (FI) technology and its most challenging components, such as the Internet of Things (IoT) [1] and the Internet of Services (IoS) [2], can provide the foundation for progress towards unified platforms that offer a variety of advanced services. Besides, the constant use of mobile devices to form interactive and participatory sensor networks, which allow users to collect, analyze and share local knowledge, can contribute to developing the smart city paradigm where the citizen is called to play an active role [3]. One of the sectors that could potentially benefit the most is tourism [4]. In such a scenario, places and objects such as sculptures, buildings, etc. can be brought into contact with users in a completely new and stimulating way [5]. In particular, data, which represents a significant added value, can be processed to enrich further the system's ability to relate man and machine. In this regard, one of the main problems is to model the awareness of the context. This problem can be solved through the Context Dimension Tree: a graph formalism representing all possible contexts [6]. The next step is to predict the possible scenarios to model the proposals to each user's needs. This further problem can be addressed through the use of Bayesian Networks [7]. Bayesian networks represent graph formalisms able to predict specific events when some variables (in our case, contextual variables) change [8].

This paper intends to propose a system based on a "Content/API Economy Platform", characterized by a strong awareness of the context. The designed system allows the content-generating actors (Institutions such as Museums, Communities or companies, and individuals) and the content user actors (the Institutions themselves, the service companies and, above all, the end-users) to operate through a platform-broker that, through the automatic composition (orchestration) of services, is able to activate in a controlled way they access and consumption of the information contained in

*Corresponding author
Email address: malombardi@unisa.it
Website: http://docenti.unisa.it/marco.lombardi
ORCID: 0000-0002-6103-594X

DOI reference number: 10.18293/JVLC2020N2-009

the Knowledge Base. In fact, the operating modes include the entire life cycle of the Knowledge Base that provides for the collection, storage, classification, and availability of the contents accessible through simple mobile applications oriented to provide tourists with richer visiting experiences [9].

In particular, through the identification and processing of the context of use in which the user operates, it is therefore essential to define flexible methods, i.e., to dynamically recommend data and services that best meet users' situational needs. When necessary, this approach can tailor the information extracted to offer the user what may be useful at a given time.

2. Background

2.1 Context Awareness

The analysis of the context in which we do something is often more important than what we are doing [10]. This concept represents why search engines are increasingly trying to understand the real meaning of individual "keywords" and the context they are placed and, therefore, the user's intent. For example, if a user is searching by typing the keyword "Japanese", is the user looking for a sushi place in the area, or is looking for language lessons?

In this regard, context-aware computing is used to indicate the use of computer technology to collect and analyze data about the reality that surrounds us. The idea is to create devices and applications that are aware of their surroundings and analyze the data to create new exciting use cases [11].

Our smartphones have been collecting contextual data for years over the network or using their sensors, such as gyroscopes, or detecting movement. They use location-based data to power many of the apps we use daily, from Google Maps to the more recent Uber [12].

Part of the challenge is that non-uniform data from heterogeneous sources can be challenging to process in a single system. In fact, data can be stored in various formats or use a different syntax that can create disambiguation problems, which could lead to misinterpretation. Fortunately, while we have access to and create more data than ever before, we also can use tools such as artificial intelligence (AI) and machine learning that can help us process this data [13].

In recent years there has also been much talk about augmented reality (AR). The resounding success of Pokémon Go has shown us how powerful this technology can be when applied correctly. In this regard, context processing will probably have a knock-on effect on the field of AR development because it provides access to new types of data that developers can exploit [14]. A consequence could be the possibility for the user to have a digital "sixth sense" available. After all, this will allow us to increase our understanding of the world around us and our possibilities, such as the quality of the air we are breathing or our speed.

The most interesting thing is that context-aware applications have led to devices that learn to know ourselves better and anticipate our moves. Google Now, for example, is specifically designed to provide information to users by predicting what they want, based on historical and contextual data [15]. As said, our smartphones are able to detect a range of information from available sensors to detect both our position (GPS) and our movements (accelerometer). To them are added wearable devices such as Fitbits, which lead, of course, to a further increase in useful information. At this point, an analysis of the data can predict when we will be hungry and how much we should eat to compensate for our activity or indicate what we probably like and what our budget is.

One of the most relevant potentials that come from contextual processing is how it can help us build artificial intelligence that speaks and can understand the environment and interact with the "senses" in a similar way to humans. Of course, we are still far from this goal. We will need a continuous joint development of machine learning and deep learning technologies to continue to progress together with context-aware computing [16]. All the data produced and available will be useless and counterproductive if we cannot process them. Context-aware technologies are based on both hardware to collect data and software to make sense of them.

Thanks to their virtual assistants, current leaders in this field are companies like Google, Apple, and Amazon. Google, Siri, and Alexa are always listening and using what they grasp to provide context-aware services. After all, if they are not listening to instructions, they cannot react to commands, so they are equipped with a certain amount of built-in contextual computing. In addition to listening to our instructions, devices like Google Home, Apple HomePod, and Amazon Echo monitor our home environment [17]. They can turn on the lights when we enter the rooms and adjust the temperature according to the weather conditions.

Ultimately, the main thing to remember is that new context-aware technologies can make our lives more comfortable, operating in an environmentally friendly way. However, it is the way we use them to offer added value in society, visible even in the long term.

2.2 Context-Aware Computing for Tourism

In the world of tourism, the advantages of using context-aware applications lead to an inevitable enhancement of cultural heritage [18]. This opportunity is linked to visitors' enormous flow, whose management is not easy and is closely related to the proper development of the tourism chain. The latter involves

both public and private operators who together are able to organize the processes necessary to achieve the objective in economic and social terms.

First of all, tourists express the need to have explicit, updated, and exhaustive references to enjoy a complete cultural heritage experience with all the necessary services (for example, local transport, catering, accommodation, information, and guides). On the other hand, the agencies and public administration want to increase the economic and social weight of the cultural heritage of their pertinence through the increase of presences and services. Finally, operators in the sector express the need for greater economic returns from interaction with the flow of visitors interested in the enjoyment of cultural heritage through engagement and relationship mechanisms dedicated to this segment of customers.

In general, the considerable amount of existing material on cultural heritage, which far exceeds the physical space available in museums or archaeological sites, and the growing interest in collections accessible to a wide audience have led institutions and professionals to increasingly adopt web-based and mobile tools to present their collections and services, meeting the needs of interested visitors [19].

Currently, with the convergence of the Internet, wireless technology, and the growing adoption of the Web as a platform for publishing information, the visitor is able to take advantage of services and material related to cultural heritage before, during, and after the visit, having different purposes and requirements at each stage. Therefore, cultural heritage exploration becomes a continuous process, starting the visit before reaching the place of interest and, ideally, never-ending. In fact, the user is able to plan and anticipate his or her travel itinerary, visit the site in person, and then revisit the places of interest using the material shared online.

It is clear that the enormous amount of information available must be filtered, customized, and contextualized to allow the individual user to access it easily. The contextualization of data and services related to cultural heritage requires a system that can model the user (for example, based on his/her interests, knowledge, and other personal characteristics), as well as contextual aspects, thus selecting the most appropriate content.

In particular, it should be remembered that museum visitors differ from each other. Their visiting to the museum is made up of the physical, personal, and socio-cultural context and aspects related to identity. Therefore, they can enormously benefit from systems that take into account personal and contextual characteristics. Moreover, visitors' behavior may not remain constant during the visit, requiring an ongoing adaptation. Besides, since tourism is a social activity, adaptation to individuals is not enough, and groups and communities also need to be shaped and supported, taking into account mutual interests and previous everyday experiences.

The challenges faced by researchers and developers of context-aware applications concern how to model and represent the user and the context of the visit and how to retrieve the available information [20]. In fact, extensive Web-based collections are difficult to identify and carry the risk of overloading users. As said, visitors are extremely heterogeneous and require different types of information and different levels of detail. Finally, in general, users of cultural heritage and tourists are often, and for a short time, visitors to a place unknown to them. On the one hand, this means that they have a constant need to find the relevant information; on the other hand, providing them with adequate answers is challenging since their interests and needs are not known from the beginning.

In this field, context-aware computing techniques can guide the selection of data and services based on the context and interests of the user or groups, protecting them from information overload [21]. Besides, contextualization and customization can be used to adapt the presentation of information on the device, thus facilitating its exploration.

However, for these purposes, heritage information must be represented, through the use of ontological models, in a format that can be interpreted by a processing system (computer, smartphone, etc.) that can be combined with the interests, preferences, and, in general, the recipient's current context [22].

3. Motivating Example

The system aims to recommend a wide range of services, which can help users during a travel experience. The system is able to help users manage the time and all the resources at his disposal in the best possible way, i.e., revealing what is around him and satisfying current and future needs.

The intention is to improve a tourist's experience and quickly project him/her into the new world in which he/she lives. It is crucial, in fact, for a person who is in a place to visit, to have the opportunity to orient themselves among the points of interest that place offers, and to know its history in order to have a conscious vision of the cultural attractions present. As soon as he or she arrives in a new place, therefore, a tourist will need to know, in general terms, the characteristics of the place and the reasons why it is worthy of interest. Through the services provided, they can immediately obtain a description and then go into all the details. Moreover, a trip does not always allow tourists to have the right time to thoroughly visit the chosen destination. In these cases, the tourists are faced with the difficult selection of the attractions that will then actually be visited. In this scenario, the tourist will be able to take advantage of the knowledge acquired

previously to visit the most important historical points and visualize the impressions that these have left in other visitors, automatically detected by, for example, Sentiment Analysis techniques. Another classic situation is to be faced with the planning of one's activities and visits; in this case, through the proposed system, tourists will be able to obtain dynamic itineraries based on their tastes and the parameters related to the current context: during the planning of the itinerary, the system must always adapt to dashboards measuring the resources that the user has made available (for example, time, budget, number of members of the visiting group, age, etc.). This system will be able, therefore, not only to perceive the whole context but also to react to it, giving appropriate answers to the user in terms of services. Imagine, for example, that the weather for the entire duration of the trip is terrible. In this case, the system has the ability to discard a priori the attractions that are outdoors and propose only "indoor services.

In this tourist scenario, the platform can be declined in mobile applications of type "Trip Designer", which build a travel itinerary by collecting from predefined folders the various steps of which the itinerary itself is composed, or through a chatbot, which maintains, through techniques of natural language processing and context recognition, a logical discourse with the user in order to respond to specific tourist needs [23]. The platform will not only have to provide a simple list of the data found on that place but will have to present them in such a way that the user can be an active part of it in order to scrutinize its past, present, and future, behaving like a modern tourist guide, also taking advantage of social networks, for years now an integral part of everyday life and containers of immense information. For all these reasons, it must enclose the set of functionalities oriented to the construction of an ecosystem to share and consult content describing the tourist/cultural heritage, exploiting, as said, a Knowledge Base.

4. System Architecture

As highlighted above, we want to propose a system for the automatic selection of services adaptive to the context and its users' needs.

The characteristics of the proposed architecture (Figure 1) mainly concern the information content that is available to end-users through the orchestration of services, proposing three different points of view:

- Representation of the Context;
- Data Management and Organization;
- Inferential Motors.

4.1 Context Representation

First of all, it is intended to convey to different categories of users, at a specific time, useful information in a given context; in practice, it is intended to create a system with a high degree of Context-Awareness. Knowledge of the context in which the user finds himself allows, in fact, to offer a wide range of services that can help the user during daily, work or private life, managing the time and resources at disposal, revealing what is around and satisfying their needs. The real-time knowledge of the context in which the user finds himself, through its representation in the form of graphs, allows, therefore, to offer highly personalized services ("tailored") able to take into account countless aspects as well as, for example, the mood of the user through an analysis of Affective Computing. For this reason, the application fields can be the most diverse: cultural heritage, tourism, e-learning, etc.

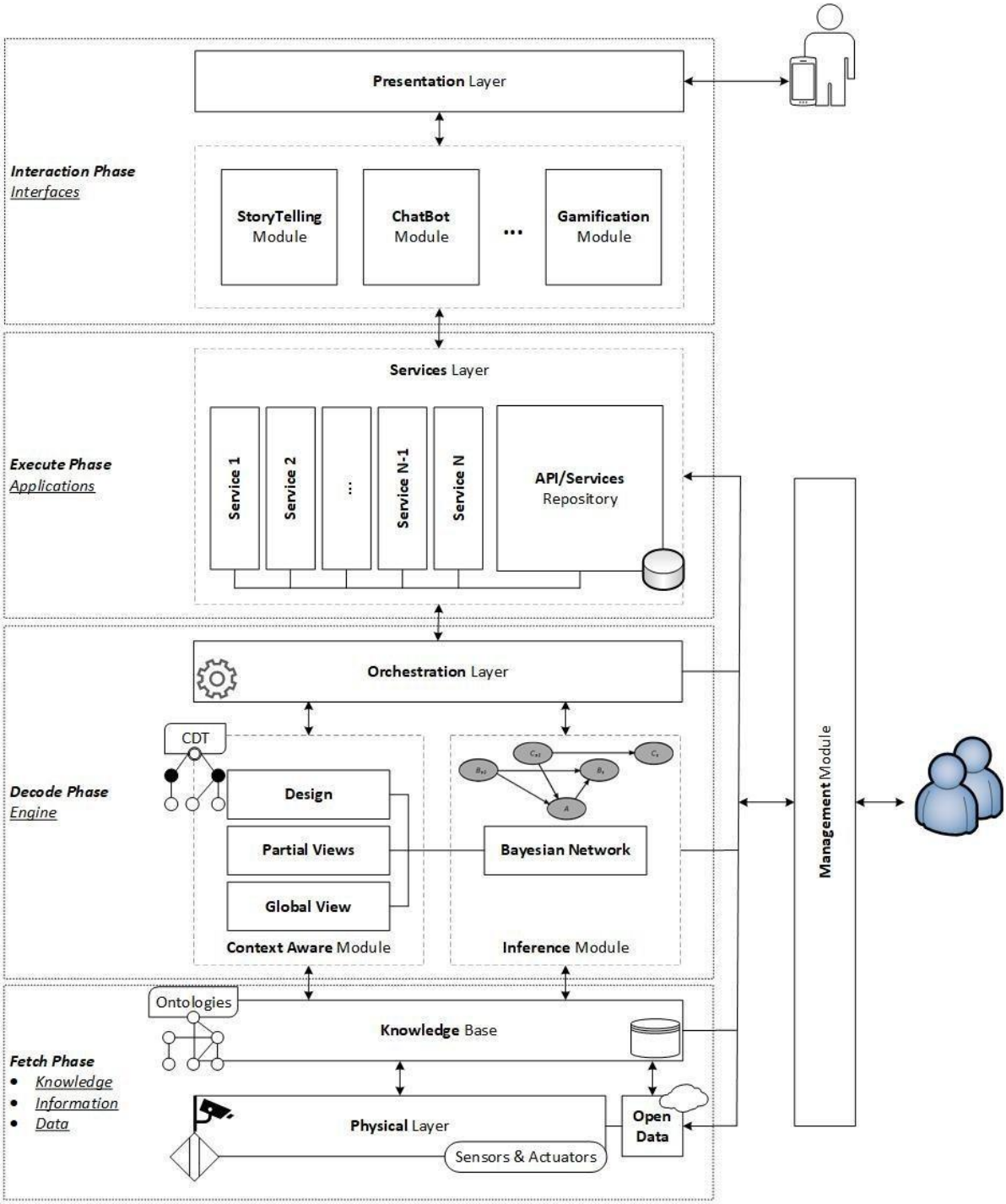
Context Awareness must be understood as a set of technical features able to give added value to services in different application segments. Context-Aware Computing applications can exploit, in our case, these features to present context information to the user or to propose an appropriate selection of actions [24]. In order to obtain a better representation of the various features, therefore, context representation formalities will be adopted, able to define, in detail, the needs of the user in the environment in which he is acting, through an approach such as: Where, Why, When, How. Everything will be declined through the state-of-the-art technologies present in the sector.

In particular, the representation of the context can occur through formal models of representation, such as the Context Dimension Tree (CDT). The latter is able to describe all the possible contexts that can be had within an application domain, through the definition of a tree consisting of a trio <r; N; A> where with r you indicate its root, with N you represent the set of nodes of which it is composed and with A the set of arcs that join these nodes. In detail, the nodes inside the CDT are divided into two categories, that of dimension nodes and that of concept nodes. The first type of node describes a possible dimension of the application domain; the second, vice versa, represents one of the possible values that a dimension can assume. The children of the root node, which constitute the top dimension, are all dimension nodes, and for each of them, a subtree may exist; the leaf nodes must be concept nodes. A dimension node can have, for children, only concept nodes and, in the same way, a concept node can have, for children, only dimension nodes. Defined, at this point, each "context element" as an assignment "dimension=value", a context will be indicated as a combination, through the use of an and, of different context elements.

Based on the use of this type of representation, the proposed methodology consists of three main phases:

- the design phase of the contexts tree, in which to identify the context elements that are significant for the application considered;
- the definition phase of partial views, in which to associate to each of them a different portion of data;
- the composition phase of the global views, in which to process the answers to the queries.

Figure 1: System Architecture



4.2 Data management and representation

In this scenario, therefore, data represent the key to building and enabling innovative services; therefore, we intend to create a Knowledge Base (KB) to collect, process and manage information in real-time. In this regard, as Knowledge Management Systems (KOS), we refer to some well-known schemes such as Taxonomies, Thesauri, or other types of vocabularies that, together with Ontologies, represent useful tools that allow modeling the reality of interest in concepts and relations between concepts. The resulting advantages are many: the use of Ontologies, for example, allows to fix a series of key concepts and definitions related to a given domain, which can be shared, providing the correct terminologies (collaborative knowledge sharing). Moreover, an ontology allows complete reuse of the knowledge encoded in it also within other ontologies or for their completion (non-redundancy of information) [25]. Electronic computers' interpretation enables the automatic treatment of knowledge, with considerable benefits (Semantic Web).

4.3 Inferential Engines

Finally, the system, designed to be in continuous operation, will have to continuously collect data from various sources and process them immediately to provide accurate services according to users and events. These, detected and analyzed, will have to be translated into facts associated with specific semantic values: it is necessary, therefore, to use an inferential engine able to conclude by applying some rules on the reported facts.

Summarizing, the need for a user can be solved in a given context by using the right services provided. The latter is characterized by innovative elements of recommendation based on the formal representation of the context, management, and organization of knowledge, inferential engines.

In particular, it is possible to define a need N_i through the following function:

$$s_i = F_{inference}(u_j, c_k)$$

Where:

$S = \{s_1, s_2 \dots, s_i\}$ represents the set of possible services that can be provided by the platform

$U = \{u_1, u_2 \dots, u_i\}$ represents the set of possible user features

$C = \{c_1, c_2 \dots, c_i\}$ represents the set of all possible contexts in a certain application domain.

5. Experimental Results

In order to provide a validation of the proposed methodology, a prototype was developed. The prototype is implemented through a hybrid mobile app and a server-side component implementation. The developed App is designed to support tourists (users) visiting Campania's coastal area (South of Italy Region). Only some user preferences and interests were considered in the first phase of methodology validation, and only the main services and points of interest have been identified. The experimental phase involved 60 volunteers aged between 21 and 55 who were unknown from the study's main purpose. The prototype was installed on the mobile device of each participant, and after an interaction phase, the system proposes a questionnaire covering several sections:

A. Presentation
B. Usability
C. Performance
D. Recommendation
E. Reliability

Each section presents two assertions associated with five possible answers according to the Likert scale: I totally disagree - TD, I disagree - D, undecided - U, I agree - A, I totally agree - TA. The answers to the questionnaire have been collected in Table 1.

Table 1: Questionnaire answers

Section	Answer				
	TD	**D**	**U**	**A**	**TA**
A	0	16	22	53	29
B	5	0	24	50	41
C	6	8	12	58	36
D	4	0	8	62	46
E	6	7	13	60	34

Table 1 shows that the users agree or strongly agree that the system provides a satisfying and reliable recommendation and contextual information and appropriate services on the site and its points of interest, meets the tourist's needs and experiences. Therefore, users show an excellent appreciation for the app: they appreciated the contents and services proposed in general.

Also, further analysis was conducted involving a smaller number of participants to evaluate the system's ability to recommend. In the first experimental phase, three pathways (P1, P2, and P3) and two activities (A1 and A2) were selected to be recommended to users. This experimental phase was divided into three steps. In the first phase, the users respond to an aptitude test. According to these tests, the users were divided into macro-groups for aptitude similarity. Subsequently, a training set was created, consisting of about 75% of the participants belonging to each macro-group. In the second phase, the training set users were able to experience the system's suggestions and interact with it. In this phase, the system could learn about the system. In the third and last phase, the users belonging to the test set group have brought to experiment with the prototype's suggestions and evaluate if the type of path

or activity suggested by the system was inherent to the context presented. The results were collected in the form of a confusion matrix in Table 2.

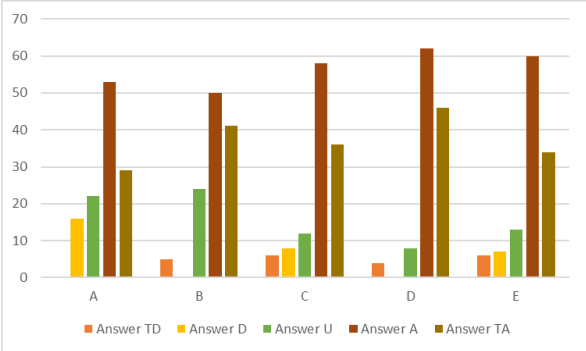
Table 2: Confusion matrix

Prediction \ Reference	P1	P2	P3	A1	A2
P1	58	8	2	7	1
P2	7	39	9	4	5
P3	0	6	45	5	7
A1	8	5	4	40	1
A2	0	6	3	4	51

Overall Accuracy : 71,69%

According to Table 2, the overall accuracy of the system is higher than 71%. This result is very encouraging and could improve over time, based on the increase of experimental data available.

Figure 2: Questionnaire answers trend



6. Conclusion and Future Works

This paper aimed to introduce a framework that can support tourists during each phase of the travel experience in the coastal area south of Italy. The system was designed to provide highly customizable and tailored services, making a tailored and unique experience. The innovation of the recommender system presented lies in the use of a high degree of context-awareness.

The proposed architecture could be used in several contexts and applications. The experimental results show that the system is able to recommend a high degree of reliability with results. In addition, the experimental campaign shows users positive feedback in-service presentation, usability, and performance shown. Future developments include improvements to the developed prototype and enlargement of the experimental campaign.

References

[1] K. Ashton, "That 'internet of things' thing," *RFID J.*, vol. 22, no. 7, pp. 97–114, 2009.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010 DOI:10.1145/1721654.1721672.

[3] G. Annunziata, F. Colace, M. De Santo, S. Lemma, and M. Lombardi, "Appoggiomarino: A context Aware app for e-citizenship," *ICEIS 2016 - Proc. 18th Int. Conf. Enterp. Inf. Syst.*, vol. 2, pp. 273–281, 2016.

[4] F. Colace, M. De Santo, M. Lombardi, and D. Santaniello, "CHARS: a Cultural Heritage Adaptive Recommender System," in *Proceedings of the 1st ACM International Workshop on Technology Enablers and Innovative Applications for Smart Cities and Communities - TESCA'19*, 2019, pp. 58–61 DOI:10.1145/3364544.3364830.

[5] F. Amato, V. Moscato, A. Picariello, F. Colace, M. De Santo, F. A. Schreiber, and L. Tanca, "Big data meets digital cultural heritage: Design and implementation of SCRABS, a smart context-aware browsing assistant for cultural environments," *J. Comput. Cult. Herit.*, 2017 DOI:10.1145/3012286.

[6] F. A. Schreiber, C. Bolchini, C. A. Curino, E. Quintarelli, and L. Tanca, "Context information for knowledge reshaping," *Int. J. Web Eng. Technol.*, 2009 DOI:10.1504/ijwet.2009.025015.

[7] F. Colace, M. Lombardi, F. Pascale, and D. Santaniello, "A multi-level approach for forecasting critical events in smart cities," in *Proceedings - DMSVIVA 2018: 24th International DMS Conference on Visualization and Visual Languages*, 2018 DOI:10.18293/DMSVIVA2018-002.

[8] F. Colace, M. De Santo, M. Lombardi, R. Mosca, and D. Santaniello, "A multilayer approach for recommending contextual learning paths," *J. Internet Serv. Inf. Secur.*, vol. 10, no. 2, pp. 91–102, 2020 DOI:10.22667/JISIS.2020.05.31.091.

[9] Y. Ioannidis, K. El Raheb, E. Toli, A. Katifori, M. Boile, and M. Mazura, "One object many stories: Introducing ICT in museums and collections through digital storytelling," in *Proceedings of the DigitalHeritage 2013 - Federating the 19th Int'l VSMM, 10th Eurographics GCH, and 2nd UNESCO Memory of the World Conferences, Plus Special Sessions fromCAA, Arqueologica 2.0 et al.*, 2013 DOI:10.1109/DigitalHeritage.2013.6743772.

[10] A. K. Dey, "Understanding and Using Context," *Pers. Ubiquitous Comput.*, vol. 5, no. 1, pp. 4–7, Feb. 2001 DOI:10.1007/s007790170019.

[11] B. Schilit, N. Adams, and R. Want, "Context-Aware Computing Applications," in *1994 First Workshop on Mobile Computing Systems and Applications*, 1994, pp. 85–90 DOI:10.1109/WMCSA.1994.16.

[12] O. Flores and L. Rayle, "How cities use regulation for innovation: the case of Uber, Lyft and Sidecar in San Francisco," *Transp. Res. Procedia*, vol. 25, pp. 3756–3768, 2017 DOI:10.1016/j.trpro.2017.05.232.

[13] R. Law, G. Li, D. K. C. Fong, and X. Han, "Tourism demand forecasting: A deep learning approach," *Ann. Tour. Res.*, vol. 75, pp. 410–423, Mar. 2019 DOI:10.1016/j.annals.2019.01.014.

[14] P. A. Rauschnabel, A. Rossmann, and M. C. tom Dieck, "An adoption framework for mobile augmented reality games: The case of Pokémon Go," *Comput. Human Behav.*, vol. 76, pp. 276–286, Nov. 2017 DOI:10.1016/j.chb.2017.07.030.

[15] S. Thakur, "Personalization for Google Now," in *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016, pp. 3–3 DOI:10.1145/2959100.2959192.

[16] D. Li, X. Chen, Z. Zhang, and K. Huang, "Learning Deep Context-Aware Features over Body and Latent Parts for Person Re-identification," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7398–7407 DOI:10.1109/CVPR.2017.782.

[17] V. Kepuska and G. Bohouta, "Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home)," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, 2018, pp. 99–103 DOI:10.1109/CCWC.2018.8301638.

[18] R. Logesh and V. Subramaniyaswamy, "Exploring Hybrid Recommender Systems for Personalized Travel Applications," 2019, pp. 535–544 DOI:10.1007/978-981-13-0617-4_52.

[19] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: A survey," *Decis. Support Syst.*, vol. 74, pp. 12–32, Jun. 2015 DOI:10.1016/j.dss.2015.03.008.

[20] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou, "Developing a context-aware electronic tourist guide," in *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '00*, 2000, pp. 17–24 DOI:10.1145/332040.332047.

[21] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "Mobile recommender systems in tourism," *J. Netw. Comput. Appl.*, vol. 39, pp. 319–333, Mar. 2014 DOI:10.1016/j.jnca.2013.04.006.

[22] F. Colace, M. De Santo, S. Lemma, M. Lombardi, A. Rossi, A. Santoriello, A. Terribile, and M. Vigorito, "How to Describe Cultural Heritage Resources in the Web 2.0 Era?," in *Proceedings - 11th International Conference on Signal-Image Technology and Internet-Based Systems, SITIS 2015*, 2016 DOI:10.1109/SITIS.2015.50.

[23] M. Casillo, F. Clarizia, G. D'Aniello, M. De Santo, M. Lombardi, and D. Santaniello, "CHAT-Bot: a Cultural Heritage Aware Teller-Bot for supporting touristic experiences," *Pattern Recognit. Lett.*, vol. 131, pp. 234–243, Jan. 2020 DOI:10.1016/j.patrec.2020.01.003.

[24] P. G. Raverdy, O. Riva, A. De La Chapelle, R. Chibout, and V. Issarny, "Efficient context-aware service discovery in multi-protocol pervasive environments," in *Proceedings - IEEE International Conference on Mobile Data Management*, 2006 DOI:10.1109/MDM.2006.78.

[25] M. Chang, G. D'Aniello, M. Gaeta, F. Orciuoli, D. Sampson, and C. Simonelli, "Building Ontology-Driven Tutoring Models for Intelligent Tutoring Systems Using Data Mining," *IEEE Access*, vol. 8, pp. 48151–48162, 2020 DOI:10.1109/ACCESS.2020.2979281.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc/

ViBERT: Visual Behavior Regression Testing

Chunying Zhao[a,*], Cong Chen[b], Kang Zhang[c] and Jun Kong[d]

[a]School of Computer Sciences, Western Illinois University, USA
[b]Independent Scholar
[c]Department of Computer Science, The University of Texas at Dallas, USA
[d]Department of Computer Science, North Dakota State University, USA

ARTICLE INFO

Article History:
Submitted 12.1.2020
Revised 12.7.2020
Second Revision 12.9.2020
Accepted 12.18.2020

Keywords:
Software Visualization
Program Comprehension
Behavior Regression Testing

ABSTRACT

Regression testing is a type of software testing that aims at identifying faults caused by code changes. Regression testing is important especially during software evolution and maintenance. As developers integrate programs or make updates to a software system, they need to make sure the changes do not adversely affect other parts of the system. Using dynamic analysis, behavioral regression testing (BERT) is one of the techniques proposed to solve the problem by re-executing test cases that target the affected area. It compares the behavior of a program before and after the changes upon certain test cases. This paper proposes *Visual BEhavioral Regression Testing (ViBERT)*, a visualization approach to comparing the behavioral differences between the new and old versions of a program in regression testing. We build a prototype called *SoftLink*, a visual environment that shows correlation/difference between two versions of a program behavior. SoftLink displays call graphs of two executions on angled parallel planes in a 3D space, and constructs correlations between them. It provides developers with an intuitive interpretation of the testing results. A case study is presented.

1. Introduction

Software visualization is defined as "the use of the crafts of typography, graphic design, animation and cinematography with modern human computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software" [38]. Visual clues such as color, shape, and metaphors ease the cognitive load of understanding software systems [7][40]. Numerous research has been proposed and various tools have been built to visualize different aspects of software systems, such as static program structures [11][17][21], dynamic program executions [8][14][16][29], software evolution[6][9], and debugging results [12].

Regression testing is an important type of software testing. During a software development and maintenance process, software may go through many changes due to system integration or software updates. Each change may introduce unwanted faults to software. Regression testing re-executes existing test cases after the source code is changed in order to determine whether the modified version has introduced regression faults into the previous working version [26]. Regression testing techniques heavily rely on the quality and sufficiency of test cases. Testers have to compromise between making thorough testing and lowering the cost. Numerous testing selection and prioritization approaches have been proposed [10][25][26][31][32][33].

Behavioral regression testing (BERT) [28][37] addresses this dilemma by identifying behavioral differences between two versions of a program through dynamic analysis. Since two consecutive revisions of a program usually do not differ significantly, BERT can reduce the number of test cases needed while achieving promising results. Behavioral regression testing relies on comparing the behavioral differences between two versions of a program through dynamic analysis to identify unforeseen side effects.

*Corresponding author
Email address: c-zhao@wiu.edu (Chunying Zhao)
congchenutd@gmail.com (Cong Chen)
kzhang@utdallas.edu (Kang Zhang)
jun.kong@ndsu.edu (Jun Kong)

DOI reference number: 10.18293/JVLC2020N2-010

Inspecting the differences between versions of program executions is tedious and error prone. To strengthen the effectiveness of BERT, this paper proposes Visual BEhavioral Regression Testing (ViBERT), a visual approach for comparing program behaviors in regression testing. We have built a semi-automatic tool called Softlink. It is a visual environment that displays and compares two or more program executions in a 3D space. It provides multiple viewpoints and directly shows the correlations between execution traces. The novelty of our approach is that it not only visually shows the differences and commonalities of two consecutive executions, but also provides a mental image of the location of the behavioral differences within the context of method calls. Our work enhances BERT with a visual representation. To our best knowledge, no studies have been conducted on comparing execution traces using 3D visualization in regression testing.

The rest of the paper is organized as follows. Section 2 illustrates a motivating example. Section 3 presents the overview of the approach. Section 4 describes how the execution traces are collected and abstracted. Section 5 shows the construction of SoftLink. Section 6 explains a case study and analyzes the results. Related work is reviewed in Section 7. Section 8 concludes the paper and presents our future work.

2. A Motivating Example

In this section, we present a motivating example to show how visualization can enhance BERT. The class *Money* is a Java package of JUnit4 library illustrating how to write unit tests with Junit [5]. As shown in Figure 1, Money.equals() is a method of Money that determines whether two monies are equal or not.

```
public boolean equals(Object anObject) {
    if (isZero())
        if (anObject instanceof IMoney)
            return ((IMoney)anObject).isZero();
    if (anObject instanceof Money) {
        Money aMoney = (Money)anObject;
        return  aMoney.currency().equals(currency())
&& amount() == aMoney.amount();
    }
    return false;
}
```

Figure 1: Original version of Money.equals().

Figure 2 shows the JUnit test cases for testing the method Money.equals().

```
public void testMoneyEquals() {
01  assertTrue ( !f12CHF.equals(null) );
    Money equalMoney = new Money(12, "CHF");
02  assertEquals (f12CHF, f12CHF);
03  assertEquals (f12CHF, equalMoney);
04  assertEquals (f12CHF.hashCode(),
                  equalMoney.hashCode() );
05  assertTrue ( !f12CHF.equals(f14CHF) );
}
```

Figure 2: Test cases in Money for Money.equals().

We deliberately remove the statement aMoney.currency().equals(currency()) && as shown in Figure 3, simulating a situation that a developer changes the code but introduces an error: the program omits checking currency when it compares two monies.

```
public boolean equals(Object anObject) {
    if (isZero())
        if (anObject instanceof IMoney)
            return ((IMoney)anObject).isZero();
    if (anObject instanceof Money) {
        Money aMoney = (Money)anObject;
        return  ~~aMoney.currency().equals(currency())
&&~~ amount() == aMoney.amount();
    }
    return false;
}
```

Figure 3: Modified version of Money.equals().

After running the test on the modified version, JUnit failed to catch the bug. The original Junit test cases are not sufficient to catch the error because the monies in the test cases have the same type of currency, such as f12CHF and f14CHF.

By visually comparing the runtime behaviors of two versions of the program, however, developers can easily identify the behavioral variations. Figure 4 correlates the behavior of two versions of the program on two 2D planes. There are observable differences in the visual presentation. The red circles drawn by hands on the left plane indicate the method invocations (currency()) executed in the original code but not executed in the modified version. With this visual hint, developers can easily locate the code affected by the modification and further check whether these behavioral variations are caused by errors or intended modifications. To further illustrate our approach, a case study has been conducted on an open-source program in Section 6. The initial results provide shows that our approach reveals the behavioral variations of the systems under study with a visual presentation.

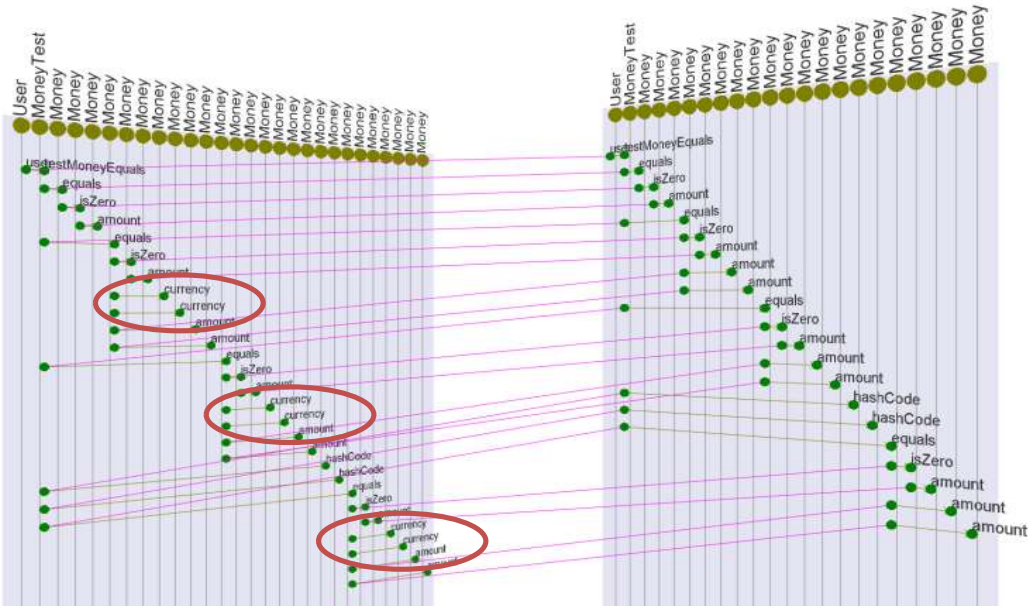


Figure 4: Visual representation of the Money executions. The left plane presents the original version, and the right one denotes the modified version. The purple correlations lines show the mapping between these two executions.

3. Visual Behavioral Regression Testing (ViBERT)

3.1 Approach Overview

Behavioral regression testing (BERT) has been used as an effective technique to identify behavioral differences between two versions of a program through dynamic analysis. Since two consecutive revisions of a program usually do not differ significantly, BERT can greatly reduce the number of test cases needed while achieving promising results. BERT typically works as follows [28]:

1) Analyze the changes between two versions and automatically generate a large number of test cases that cover the changed parts of the code.
2) Run the generated test cases on the old and new versions of the code and identify differences in the tests' outputs.
3) Analyze the identified differences and presenting them to the developer.

BERT analyzes behavioral differences by comparing the program outcomes. SoftLink complements BERT by visualizing behavioral variations. SoftLink can work seamlessly with existing BERT tools, and display the correlations between consecutive versions. As an enhancement to BERT, ViBERT works in the following steps:

1) Insert AspectJ instrumentations to the test suite automatically generated in Step 1 of BERT that focuses on the changed parts of the program.
2) Run Step 2 of BERT and generate traces for the two executions to be compared.
3) Use SoftLink to visualize the correlations between two versions of program executions and highlight their differences.

3.2 Design Characteristics

As a software visualization tool, SoftLink is specifically tailored to correlation visualization. SoftLink visualizes abstracted call graphs on 2D planes in a 3D space. SoftLink takes advantage of the benefits of angled and paralleled views. As the viewpoint is changed, the arrangements of planes can be dynamically updated accordingly, in a similar fashion as a camera model. Planes with corresponding correlations interesting to the user always face the user as depicted in Figure 5.

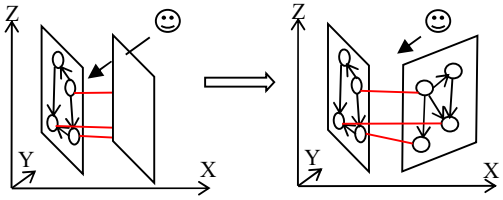


Figure 5: Auto-orienting views.

We design the visual features of SoftLink following the general functional requirements proposed by Kienle and Müller [23]:

- **Views** (linked static views and dynamically synchronized views): SoftLink incorporates three views: a 3D correlation view, a modulation view, and a source code view. These views are linked such that the change of one view automatically triggers the changes of the others. Moreover, these views, especially the 3D correlation view, are

dynamically synchronized with the underlying data.

- **Abstraction**: To effectively visualize complex software systems, a visualization tool should support adjustable granularities (i.e. abstraction levels) and provide sufficiently detailed information on demand. We use a multi-level abstraction on execution traces to enable in-dept exploration of program. In SoftLink, nested method calls can be folded or unfolded corresponding to the change of the abstraction level.
- **Search and Code Proximity**: Finding text strings in the source code corresponding to objects in the visual representation is considered "absolutely essential" [23]. SoftLink provides a query function with a search bar, where users can easily locate source code to their interests.
- **Automatic Layout**: SoftLink uses a multi-plane presentation to visualize multiple executions. It automatically displays execution planes in the 3D space. Planes are dynamically angled towards the user so that both individual executions and their correlations have the best exposure.
- **History/Undo**: SoftLink has an interactive interface that allows the user to click on visual objects while navigating in the 3D space. Iteratively, upon each click, a new nested plane visualizing the detailed information is popped up. All the upper-level planes are kept on the screen to show the browsing history.

4. Execution Traces

4.1 Execution Trace Collection

Obtaining execution traces is the first step to correlate executions. We choose AspectJ[2], a Java implementation of aspect-oriented programming, to intercept program execution metadata, because it can non-intrusively extract runtime traces with a high level of flexibility and expressiveness.

Using the following aspect in Figure 6, we record each method call's signature along with the name of the object it belongs to and the time the program enters and leaves the method.

```
public aspect Trace {
  pointcut allCalls() : execution( * *.*(..));
  before() : allCalls() {
    String signature =
thisJoinPointStaticPart.getSignature().toShortString();
    if(!signature.isEmpty()) {
      String log = "-> "+ signature+ "$" +
Thread.currentThread().getName()+ "*" +
System.currentTimeMillis() + "$";
      System.out.println(log);
    }
  }
  after() : allCalls() {
    String signature =
thisJoinPointStaticPart.getSignature().toShortString();
    if(!signature.isEmpty()) {
      String log = "<- "+ signature+ "$" +
Thread.currentThread().getName()+ "*" +
System.currentTimeMillis() + "$";
      System.out.println(log);
    }
  }
}
```

Figure 6: Definition of Aspect

The plain-text trace log captured by this aspect is then imported to SoftLink, and automatically transformed to call graphs specified in GraphML [3], an XML-based graph presentation. In Softlink, we enhance our previous work on program abstraction [43] and built an Abstracer to perform such transformation.

4.2 Execution Trace Representation and Abstraction

Execution traces need to be properly represented and abstracted before being analyzed as otherwise the user can be misled by partial information or overwhelmed by too much trivial information. Proper abstraction at various granularities makes it possible to display a large volume of program data on limited visual space. When comparing program executions, we identify equivalent substructures in two abstracted call graphs. We represent the call graph *G(N,E)* in a tree structure that consists of multiple caller-callee chains built from the GraphML runtime trace.

Definition 1: A call graph *G(N,E)* is a directed node-link graph, where the set of nodes *N* denote methods and the set of edges *E* represent method invocations.

Each edge directs from a caller to a callee. Each method invocation is annotated with two parameters: the *depth* and the *length* of the call chain. The depth here refers to the depth of the call chain through which we adjust the granularity of the visualization. The length of a call chain is defined as the number of nodes in the path from the root to the leaf in a call graph, which is used to prune short call chains in Abstracer. Each directed edge is annotated with the number of method call repetitions. For instance, there are three call chains in the call graph of Figure 7: *a-b*, *a-c-d*, and *a-c-e*. The lengths of each call chain are 2, 3, and 3, respectively. The depth of each method is as follows: θ_{depth}(a) = 1, θ_{depth}(b) =

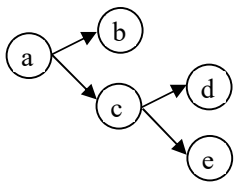


Figure 7: An example for abstraction

$\theta_{depth}(c) = 2$, $\theta_{depth}(d) = \theta_{depth}(e) = 3$.

Abstracer, the abstraction engine integrated into SoftLink, is used to remove less significant information not to be shown in the graphical representation, such as method calls that contribute little to the comprehension of program behavior. We consider three criteria for execution abstraction:

- **Continuous repetitions**. Continuously repeated method invocations can be collapsed to one occurrence. Such repetitions mostly manifest themselves as loops. For instance, in a sequence of method calls *EABCABCF* (a letter represents a method call), the call sequence *ABC* is considered duplicated. Thus, only one occurrence of *ABC* is shown in the call graph. We can label the corresponding edge in the call graph with the number of repetitions. Noncontiguous repetitions are not collapsed because they may belong to different abstraction levels.
- **Depth of methods in a call chain**. This type of abstraction relies on the depth threshold θ_{depth} (an integer specified by the user). Methods whose nesting depths in the call chain are deeper than this threshold can be collapsed. For instance, given θ_{depth} =3, the methods with a depth of 4 or more are collapsed, and not shown in the abstracted scenario. Low-level methods provide detailed information for high-level abstract events and can be unfolded if the user lowers the abstraction level.
- **Short call chains**. A long call chain including more method invocations may represent a significant function of a program. SoftLink uses parameter θ_{length} as a threshold to specify the minimum length of call chains. Method invocations with call chains shorter than θ_{length} are pruned. SoftLink abstracts the call graph by traversing it and collapse methods according to the customizable parameters θ_{depth} and θ_{length}.

5. SOFTLINK

5.1 Overview

Figure 8 shows the interface of SoftLink that includes three views: a 3D correlation view, a modulation view, and a source code view. The controls of SoftLink are on the menu bar. The user can use the *file* menu to import trace logs and specify the number of executions to be correlated. The *action* menu includes commands for specifying trace abstraction levels and constructing correlations. SoftLink first loads the selected plain-text trace files and transforms the files into call graphs in GraphML. It trims the call graphs based on the abstraction parameters set by the user. Call graphs are displayed on individual 2D planes, similar to sequence

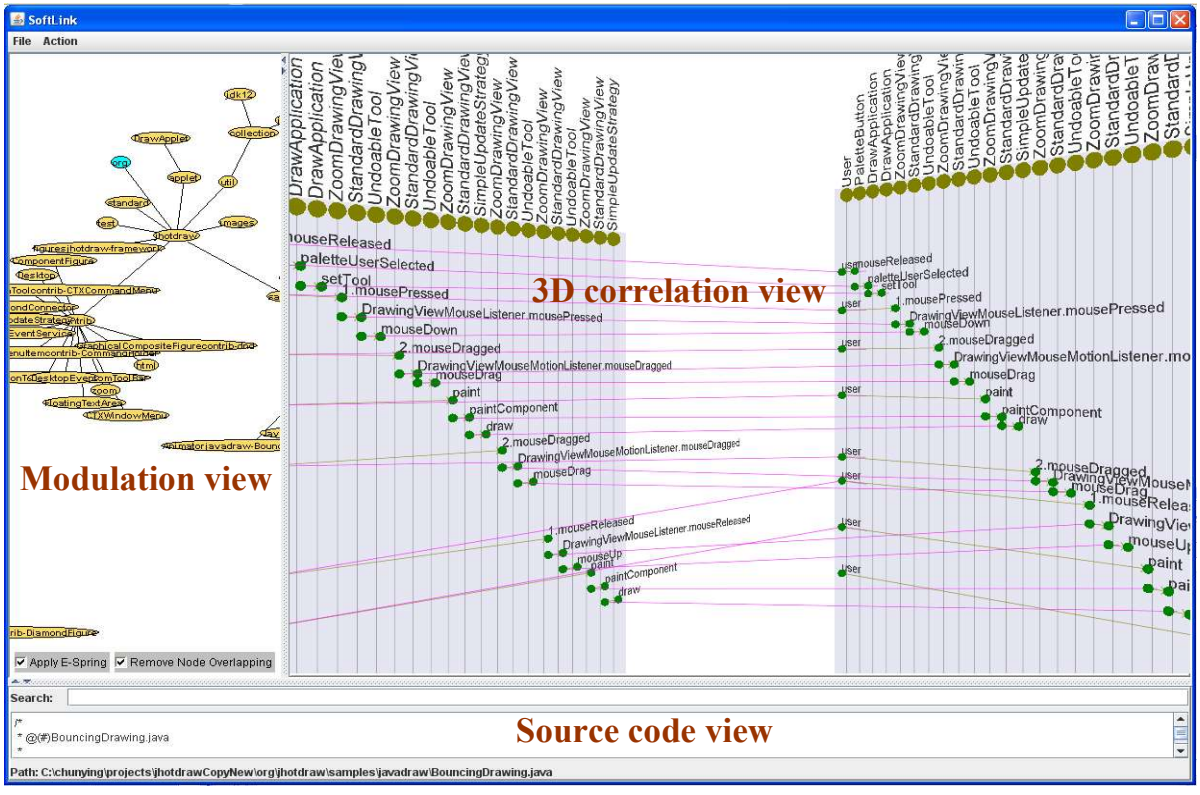


Figure 8: Three views of SoftLink.

diagrams. Then, the correlations are visualized in the 3D correlation view. The 3D scene is automatically rendered when the user changes the abstraction level.

5.2 Views in SoftLink

The 3D correlation view is built using Java3D. Each brown sphere represents an object. Green spheres represent methods. Horizontal green lines indicate the calling relationship between methods. Purple lines represent the correlations between two executions. These colors are selected to achieve some contrast against the background. This 3D scene provides users with multiple viewpoints to observe the relationship between executions. Users can choose to observe the differences or commonalities.

The modulation view shows the hierarchical packaging structure of the program in a force-directed layout using the E-spring Algorithm [24]. High-level organizations of system modules represent the composition of a system and are commonly visualized using the tree structure in a node-link graphical format. Each sub-package is a child node of its parent package, while tree leaves represent files. This modulation view gives developers an overview of the program structure.

The source code view provides a fast access to methods in the source file corresponding to the visual entity that the user is interested in. Having spotted desired information in the 3D visual representation, the user might need to check the corresponding source code. The source code view of SoftLink is synchronized with the visual representation in the 3D correlation view by highlighting the queried method in red. The file path of the searched method is shown in the status bar in the source code view.

5.3 Iterative Multi-level Nested Visualization

5.3.1 Zooming and Rotatable Scene

SoftLink provides efficient interaction and navigation capabilities. In the current implementation, mouse is used for picking and rotating individual planes and visual objects in the 3D correlation view. SoftLink allows the user to move or rotate each single plane to any angle around any axis in 3D space. The keyboard is used to control the entire 3D scene, such as zooming, rotating, and moving the user's viewpoint. To provide customizable views, when the viewpoint moves, each individual plane in the scene can be adjusted accordingly to the viewpoint.

5.3.2 Iterative Multi-level Nested Visualization

Program execution is hard to visualize if all the method invocations need to be displayed. Even if only a portion of a software system is executed, the collected

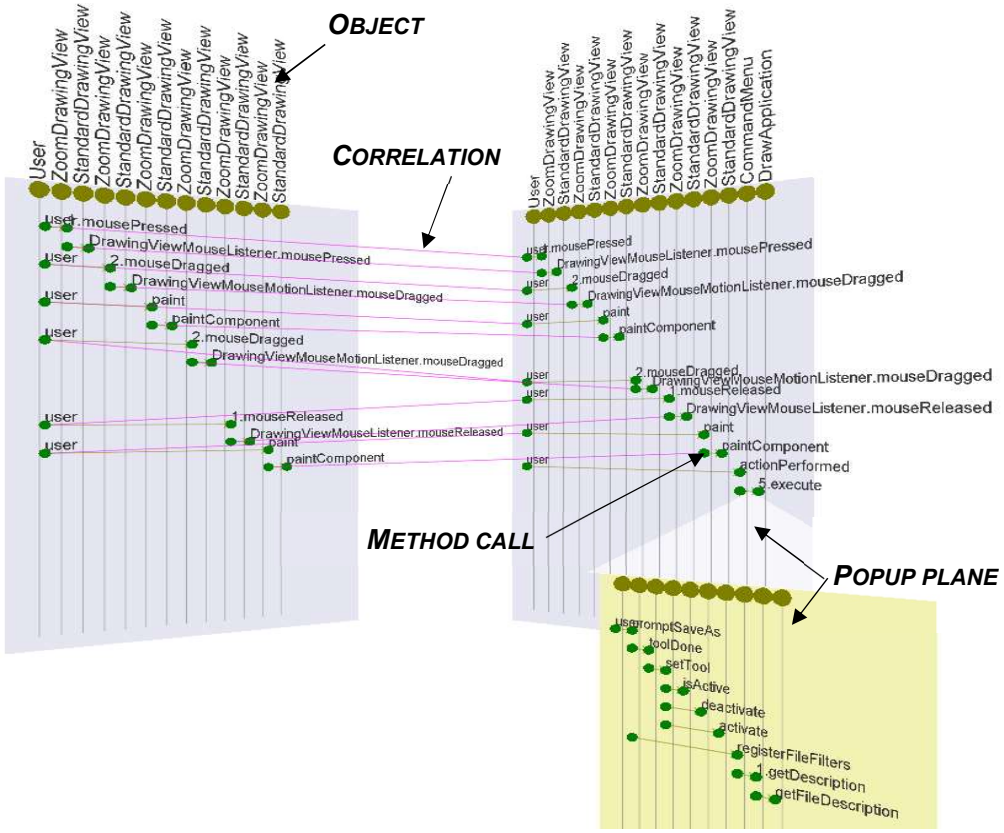


Figure 9: Visual representation of correlations between two executions.

traces can be millions of lines, making it incomprehensible. To address this limitation, SoftLink utilizes multi-level abstraction and iterative drill-down visualization as in Figure 9. When the user clicks a visual object representing a method, the nested interactions within that method are shown in a popup plane attached to the clicked method. Iteratively, the visual objects on the newly popped plane can also be unfolded upon the user's click. By drilling down through multiple planes, the user can get more detailed information. This capability is particularly suitable for a system equipped with an eye-tracker [22].

6. Case Study

We applied ViBERT on an industrial software package, *Joda-Time* [4], a Java date and time library. Joda-Time 1.6.2 has about 4615 classes in the source code, and 4080 classes in the testing code. We select a number of real bugs detected and fixed in the development process of Joda-Time.

Both BERT and ViBERT focus on the changes between two adjacent revisions, we select two revisions of Joda-Time to simulate the regression testing. Suppose revision r_i fixes a bug in revision r_j, we can interpret that revision r_j introduces the bug in *revision* r_i, and use this bug to simulate a real regression fault. We apply ViBERT to identify the regression faults in the changes between revisions of the program.

The *Subversion* repository[1] of Joda-Time contains about 1610 revisions. We search the history for the revisions that have fixed bugs in previous revisions. 156 version pairs <r_i, r_j> are found, where revision r_i fixes certain bugs in revision r_j. Finding regression faults in a software's history is time consuming, requiring a manual process:

(1) Search the revision history for a bug that has been fixed.

(2) Locate the revision and the source code where the bug first appears.

(3) Examine whether the interface of the source code has been changed between the revision that introduces the bug and the revision before it. If the interface is not changed, then the bug is considered a regression fault that was introduced by the new revision.

The selected revision pair is <r1576, r1577>. Revision r1577 fixes a bug in the method *AbstractDuration.toString()* in revision r1576. This method produces wrong output for negative inputs. Putting these two revisions in a regression testing setting, we take revision r1577 as the old version without the bug, and revision r1576 as the new version that introduces bugs.

To catch the bug using regression testing, developers first study the changes the new version has made to the source code, and then create test cases targeting the changes. A test suite containing nine JUnit test cases for the method *AbstractDuration.toString()* is defined as shown in Figure 10:

```
public void testToString() {
01  assertEquals("PT0S",
                  new Duration(0L).toString());
02  assertEquals("PT10S",
                 new Duration(10000L).toString());
03  assertEquals("PT1S",
                  new Duration(1000L).toString());
04  assertEquals("PT12.345S",
                 new Duration(12345L).toString());
05  assertEquals("PT-12.345S",
                new Duration(-12345L).toString());
06  assertEquals("PT-1.123S",
                 new Duration(-1123L).toString());
07  assertEquals("PT-0.123S",
                 new Duration(-123L).toString());
08  assertEquals("PT-0.012S",
                  new Duration(-12L).toString());
09  assertEquals("PT-0.001S",
                  new Duration(-1L).toString());
}
```

Figure 10: Test cases in Joda-Time for Duration.toString().

We first run the existing JUnit test suite on the new version of the program, the test suspends at test case 07, indicating that the actual output is not expected. By observing the changes that the new version has made to the source code, we can conclude that the program behavior starts to differ when the length of the output is greater than 8 (or 7 if the output is a positive number). In the given test suite, however, starting from test case 04, the lengths of the expected outputs are all greater than 8. The program actually behaves differently since test case 04. Therefore, although running the existing test suite can eventually capture the bug, it could have revealed the change earlier (from test case 04 instead of test case 07).

By exploring the behavioral variations of these two revisions, ViBERT intends to detect this potential problem, and alerts developers with visual hints. We run the test cases on both versions and compare their differences. As an enhancement to behavioral regression testing, we use visual representations to show behavioral difference. We obtain the execution traces by embedding AspectJ instrumentations into JUnit testing code. Then, ViBERT visualizes the correlations between the two executions. Figure 11 shows the result in SoftLink. The left plane represents revision r1577, and the right one represents r1576. The numbers correspond to the test cases in Figure 10.

The left plane successfully runs all the test cases. The right plane shows only 7 test cases, because the test stops at test case 07. We notice that since test case 04, two executions exhibit different behaviors due to the changes to the code. The method calls to "*appendPaddedInteger*" in the right panel (revision: r1576) do not exist in the left panel (revision: r1577). Via visual inspection, developers can identify the

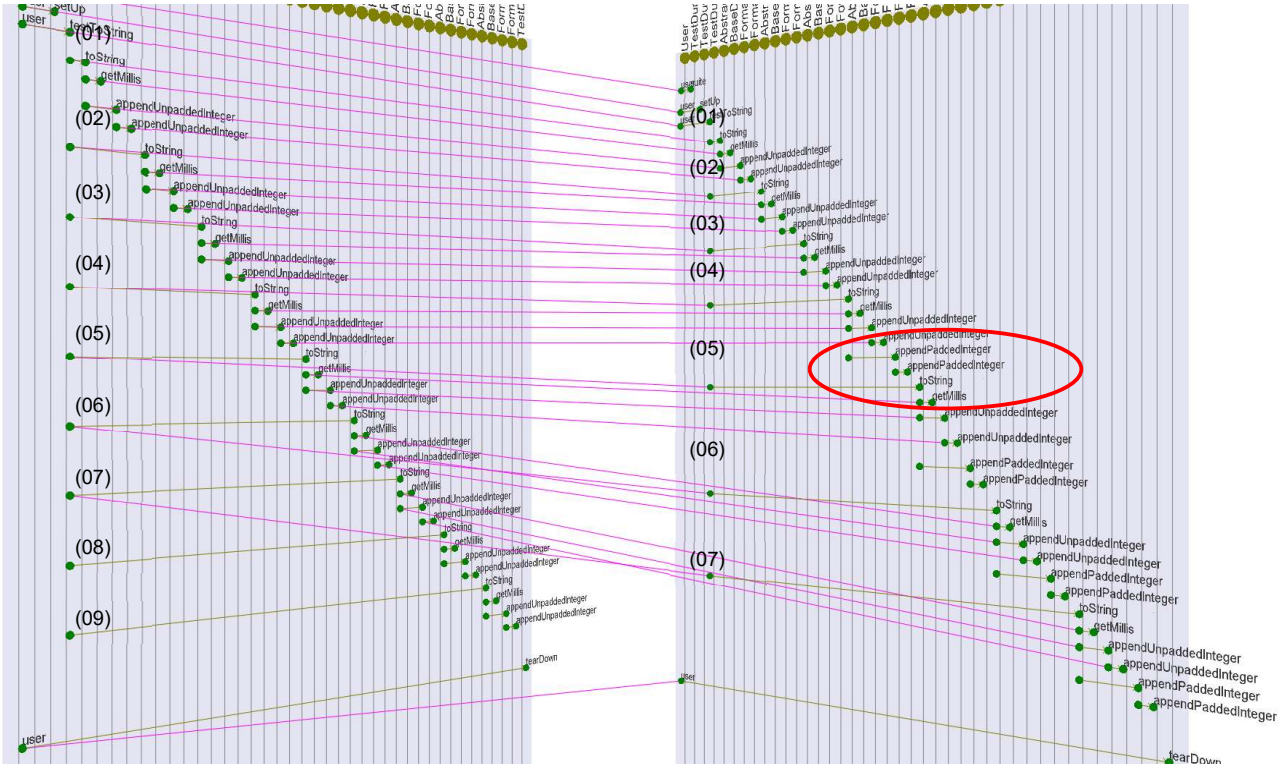


Figure 11: ViBERT on revision pair <r1577, r1576> of Joda-Time. The left plane visualizes revision r1577, and the right one presents r1576.

influence of the changes to the program behavior, and further analyze whether those changes introduce new errors.

As shown in the case study, ViBERT shows the differences between two executions using visual representations. It displays where are the differences and the context of the differences in the execution. In this experiment, we use one revision pair as one example. Other revision pair can be compared in a similar fashion. In software testing, there are many test coverage criteria and metrics. It is worthwhile to note that as Behavior Regression Testing focuses only on comparing method invocations in program executions, not all types of program errors can be identified by behavior regression testing.

7. Related Work

7.1 Program Behavior Comprehension

Numerous researchers have focused on visualizing program executions. Comprehensive surveys of dynamic analysis and software visualization [15][35] are available. Traditionally program behaviors are represented as node-link diagrams in a two-dimensional space. Examples include UML sequence diagrams [10], space-time diagrams, and call graphs [42].

Researchers utilize essential visual elements such as color, shape, and a variety of visual layouts to represent software information. Popular layouts include trees (e.g. tree map [34][39]), tables, graphs and diagrams. TraceVis [30] visualizes executed program instructions by sequentially displaying microprocessor instructions in a 2D plane. It supports queries, different levels of zooming, and annotations on colorful blocks. GAMMATELLA [27] visualizes executions in three levels in 2D: a file level represented in a miniaturized view, a system level using a tree map, and a statement level. MetropoIJS [34] visualizes static and dynamic aspects of largescale program written in Javascript with Treemaps. These approaches, however, focus on the visualization of single execution scenario and do not support a comparison of different program executions. Our study complements previous research by applying existing successful layouts on individual 2D planes in SoftLink.

Apart from 2D visualization, more 3D software visualization environments are built through virtual realities. Metaphors such as cities were used to represent software systems [11][36]. Fittkau *et al*. [20] designed controlled experiments to compare the trace visualization tools EXTRAVIS [14] and ExplorViz in program comprehension tasks. EXTRAVIS uses circular bundling and a massive sequence view, and ExplorViz uses the city metaphors. Scalability in software visualization are commonly addressed by using multiple levels of abstraction [19][41].

7.2 Regression Testing and Visualization

Regression testing aims at uncovering new errors after changes are made to a software system. The increasing size of software systems makes thorough

regression testing a costly endeavor. In addition to traditional test selection and prioritization techniques, researchers have applied visual analytics to regression testing. Engström et al. [18] utilize a heat map (mosaic visualization) to show test history and test covered items. Chen and Ince [13] design a tabular visual representation of regression test results. Different colors are assigned to the blocks on the table and fisheye enlarges the rows of users' interest.

BERT [28][37] is a differential testing technique that identifies behavioral differences between two versions of a program through automatically generated test cases and dynamic analysis. Different from previous testing work, ViBERT compares dynamic program behavior and complements the BERT technique with a visual tool SoftLink.

8. Conclusion and Future work

Regression testing aims at identifying unnoticed faults caused by changes to software. Behavioral regression testing uses dynamic analysis to compare new and old versions of a program in regression testing. This paper has proposed ViBERT, a visual approach to comparing program behavior. Specifically, we had built a 3D environment that allows developers to view the correlations and differences between two versions of program executions. In contrast to other visualization tools, our approach focuses on consecutive behavior comparison. It helps users to interpret the behavioral differences within the context of the executions.

Our future work is to conduct a usability study and gather more feedbacks from users. We also plan to integrate this environment with popular IDEs, such as Eclipse and IntelliJ. More experiments on larger software systems will also be conducted. Another possible extension is that the viewpoint-oriented representation can be enhanced with an eye tracker. The position of the pupil in the eye-tracking controller screen is mapped to that in the visual space. We can use the eye tracker to capture the user's visual focus, and as the viewer's focus moves, the orientations of planes will be automatically updated accordingly.

References

[1] Apache Subversion. http://subversion.apache.org/

[2] AspectJ.https://www.eclipse.org/aspectj/

[3] GraphML. http://graphml.graphdrawing.org/

[4] Joda-Time. http://joda-time.sourceforge.net/

[5] JUnit. http://www.junit.org/

[6] Alexandru C. V., Proksch S., Behnamghader P. and Gall H. C., "Evo Clocks: Software Evolution at a Glance," Working Conference on Software Visualization (VISSOFT), 2019, pp. 12-22.

[7] Ball T. and Eick S. G., "Software Visualization in the Large". Computer, vol. 29,1996, pp. 33-43.

[8] Beck F., Siddiqui H. A., Bergel A. and Weiskopf D., "Method Execution Reports: Generating Text and Visualization to Describe Program Behavior". IEEE Working Conference on Software Visualization, 2017, pp. 1-10.

[9] Behnamghader P., Alfayez R., Srisopha K., and Boehm B., "Towards Better Understanding of Software Quality Evolution through Commit-Impact Analysis". IEEE International Conference on Software Quality, Reliability and Security (QRS), 2017, pp. 251-262.

[10] Briand L.C., Labiche Y., He S., "Automating Regression Test Selection based on UML Designs". Information and Software Technology, Vol. 51, No.1, 2009, pp. 16-30.

[11] Capece N., Erra U., Romano S., and Scanniello G., "Visualising a Software System as a City Through Virtual Reality". Augmented Reality, Virtual Reality, and Computer Graphics, 2017, pp. 319–327.

[12] Castro D. and Schots M., "Analysis of Test Log Information through Interactive Visualizations". International Conference on Program Comprehension, 2018, pp. 156-166.

[13] Chen R. and Ince T., "Visualizing Regression Test Results". http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=B6B26126F10004A55199CC40E57E896D?doi=10.1.1.366.3623&rep=rep1&type=pdf.

[14] Cornelissen B., Holten D., Zaidman A., Moonen L., Wijk J. J. v., and Deursen A. V., "Understanding Execution Traces Using Massive Sequence and Circular Bundle Views". IEEE International Conference on Program Comprehension, 2007, pp. 49-58.

[15] Cornelissen B., Zaidman A., Deursen A. V., and Moonen L., "A Systematic Survey of Program Comprehension through Dynamic Analysis". IEEE transaction on Software engineering, Vol 35, No. 5, 2009, pp. 684-702.

[16] Cornelissen B., Zaidman A., and Deursen A. V., "A Controlled Experiment for Program Comprehension through Trace Visualization". IEEE Transactions on Software Engineering, Vol.37, No.3, 2011, pp.341-355.

[17] Eick S. C., Steffen J. L. and Sumner, E. E., "Seesoft - a tool for Visualizing Line Oriented Software Statistics". IEEE Transactions on Software Engineering, Vol. 18, No. 11,1992, pp. 957-968.

[18] Engström E., Mantylä M., Runeson P. and Borg M., "Supporting Regression Test Scoping with Visual Analytics". IEEE 7th International Conference on Software Testing, Verification and Validation, 2014, pp. 283-292.

[19] Feng Y., Dreef K., Jones J. A., and Deursen A. V., "Hierarchical Abstraction of Execution Traces for Program Comprehension". International Conference on Program Comprehension, 2018, pp. 86-96.

[20] Fittkau F., Finke S., Hasselbring W. and Waller J., "Comparing Trace Visualizations for Program Comprehension through Controlled Experiments". IEEE International Conference on Program Comprehension, 2015, pp. 266-276.

[21] Lanza M. and Ducasse S., "Polymetric views - a lightweight visual approach to reverse engineering". IEEE Transactions on Software Engineering, Sep. 2003, Vol. 29, No. 9, pp. 782–795.

[22] Jbara A. and Feitelson D. G., "How Programmers Read Regular Code: A Controlled Experiment Using Eye Tracking". IEEE International Conference on Program Comprehension, 2015, pp. 244-254.

[23] Kienle H. M. and Müller H. A., "Requirements of Software Visualization Tools: A Literature Survey". 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, 2007, pp. 2-9.

[24] Kumar P., Zhang K., Wang Y., "Visualization of Clustered Directed Acyclic Graphs without Node Overlapping". 12th International Conference on Information Visualization, 2008, pp. 38-43.

[25] Mostafa S., Wang X., Xie T., "PerfRanker: Prioritization of Performance Regression Tests for Collection-intensive Software". International Symposium on Software Testing and Analysis, 2017, pp. 23-34.

[26] Nardo D. D., Alshahwan N., Briand L. C., Labiche Y., "Coverage-based Regression Test Case Selection, Minimization and Prioritization: a Case Study on an Industrial System".

Software Testing, Verification, and Reliability, Vol 25, No.4, 2015, pp. 371-396.

[27] Orso A., Jones J. A., Harrold M. J., and Stasko J., "GAMMATELLA: Visualization of Program-execution Data for Deployed Software". 26th International Conference on Software Engineering, pp. 699-700, 2004.

[28] Orso A. and Xie T., "BERT: BEhavioral Regression Testing". International Workshop on Dynamic Analysis, pp. 36-42, 2008.

[29] Reiss S. P., "Visual Representations of Executing Programs". Journal of Visual Languages and Computing, vol. 18, pp. 126-148, 2007.

[30] Roberts J. and Zilles C., "TraceVis: An Execution Trace Visualization Tool". 1st Workshop on Modeling, Benchmarking and Simulation, 2005, pp. 31-38.

[31] Rothermel G., Harrold M. J., "Analyzing Regression Test Selection Techniques". IEEE Transactions on Software Engineering, Vol.22, No.8,1996, pp. 529-551.

[32] Rothermel G., Elbaum S. G., Malishevsky A. G., Kallakuri P., Qiu X., "On Test Suite Composition and Cost-effective Regression Testing". ACM Transaction on Software Engineering and Methodology, Vol. 13, No.3, 2004, pp.277-331.

[33] Rothermel G., "Improving Regression Testing in Continuous Integration Development Environments". keynote at ESEC/SIGSOFT FSE 2018.

[34] Scarsbrook J. D., K.L. KO R., Rogers B., Brainbriage D., "MetropolJS: Visualizing and Debugging Large-Scale JavaScript Program Structure with Treemaps". International Conference on Program Comprehension, 2018, pp.389-392.

[35] Teyseyre A. R. and Campo M. R., "An Overview of 3D Software Visualization". IEEE Transactions on Visualization and Computer Graphics, Vol. 15, No. 1, 2009, pp. 87-105.

[36] Wettel R. and Lanza M., "Codecity: 3D Visualization of Largescale Software". Companion of 30th International Conference on Software Engineering, 2008, pp. 921–922.

[37] Wei J., Orso A., and Xie T., "Automated Behavioral Regression Testing". 3rd International Conference on Software Testing, Verification and Validation, 2010, pp. 137-146.

[38] Stasko J. T., Brown M. H., Domingue J. B., Price B. A., Software Visualization: Programming as a Multimedia Experience, MIT Press, 1998.

[39] Yang Y.L., Zhang K., Wang J.R., and Nguyen Q.V., "Cabinet Tree: An Orthogonal Enclosure Approach to Visualizing and Exploring Big Data". Journal of Big Data, Springer, 2:15, December 2015.

[40] Zhang K., ed., Software Visualization - From Theory to Practice. Boston: Kluwer Academic Publishers, 2003.

[41] Zhao C., Zhang K., Hao J., and Wong W. E., "Visualizing Multiple Program Executions to Assist Behavior Verification". 3rd IEEE International Conference on Secure Software Integration and Reliability Improvement, 2009, pp. 113-122.

[42] Zhao C., Kong J. and Zhang K., "Program Behavior Discovery and Verification: A Graph Grammar Approach". IEEE Transactions on Software Engineering, Vol. 36, No. 3, 2010, pp. 431-448.

[43] Zhao C., Zhang K., and Lei Y., "Abstraction of Multiple Executions of Object-oriented Programs". ACM symposium on Applied Computing, 2009, pp. 549-550.

Journal of Visual Language and Computing

Volume 2020, Number 2