Zbornik 21. mednarodne multikonference

INFORMACIJSKA DRUŽBA - IS 2018

Zvezek G

Proceedings of the 21st International Multiconference

INFORMATION SOCIETY - IS 2018

Volume G

Sodelovanje, programska oprema in storitve v informacijski družbi

Collaboration, Software and Services in Information Society

Uredil / Edited by
Marjan Heričko

http://is.ijs.si

8.–12. oktober 2018 / 8–12 October 2018
Ljubljana, Slovenia

Zbornik 21. mednarodne multikonference

INFORMACIJSKA DRUŽBA – IS 2018

Zvezek G

Proceedings of the 21st International Multiconference

INFORMATION SOCIETY – IS 2018

Volume G

Sodelovanje, programska oprema in storitve v informacijski družbi
Collaboration, Software and Services in Information Society

Uredil / Edited by

Marjan Heričko

http://is.ijs.si

8.–12. oktober 2018 / 8–12 October 2018
Ljubljana, Slovenia

Urednik:

Marjan Heričko
University of Maribor
Faculty of Electrical Engineering and Computer Science

Založnik: Institut »Jožef Stefan«, Ljubljana
Priprava zbornika: Mitja Lasič, Vesna Lasič, Lana Zemljak
Oblikovanje naslovnice: Vesna Lasič

Dostop do e-publikacije:
http://library.ijs.si/Stacks/Proceedings/InformationSociety

Ljubljana, oktober 2018

Informacijska družba
ISSN 2630-371X

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani
COBISS.SI-ID=21853462
ISBN 978-961-264-141-2 (pdf)

PREDGOVOR MULTIKONFERENCI INFORMACIJSKA DRUŽBA 2018

Multikonferenca Informacijska družba (http://is.ijs.si) je z enaindvajseto zaporedno prireditvijo osrednji srednjeevropski dogodek na področju informacijske družbe, računalništva in informatike. Letošnja prireditev se ponovno odvija na več lokacijah, osrednji dogodki pa so na Institutu »Jožef Stefan«.

Informacijska družba, znanje in umetna inteligenca so še naprej nosilni koncepti človeške civilizacije. Se bo neverjetna rast nadaljevala in nas ponesla v novo civilizacijsko obdobje ali pa se bo rast upočasnila in začela stagnirati? Bosta IKT in zlasti umetna inteligenca omogočila nadaljnji razcvet civilizacije ali pa bodo demografske, družbene, medčloveške in okoljske težave povzročile zadušitev rasti? Čedalje več pokazateljev kaže v oba ekstrema – da prehajamo v naslednje civilizacijsko obdobje, hkrati pa so notranji in zunanji konflikti sodobne družbe čedalje težje obvladljivi.

Letos smo v multikonferenco povezali 11 odličnih neodvisnih konferenc. Predstavljenih bo 215 predstavitev, povzetkov in referatov v okviru samostojnih konferenc in delavnic. Prireditev bodo spremljale okrogle mize in razprave ter posebni dogodki, kot je svečana podelitev nagrad. Izbrani prispevki bodo izšli tudi v posebni številki revije Informatica, ki se ponaša z 42-letno tradicijo odlične znanstvene revije.

Multikonferenco Informacijska družba 2018 sestavljajo naslednje samostojne konference:

- Slovenska konferenca o umetni inteligenci
- Kognitivna znanost
- Odkrivanje znanja in podatkovna skladišča – SiKDD
- Mednarodna konferenca o visokozmogljivi optimizaciji v industriji, HPOI
- Delavnica AS-IT-IC
- Soočanje z demografskimi izzivi
- Sodelovanje, programska oprema in storitve v informacijski družbi
- Delavnica za elektronsko in mobilno zdravje ter pametna mesta
- Vzgoja in izobraževanje v informacijski družbi
- 5. študentska računalniška konferenca
- Mednarodna konferenca o prenosu tehnologij (ITTC)

Soorganizatorji in podporniki konference so različne raziskovalne institucije in združenja, med njimi tudi ACM Slovenija, Slovensko društvo za umetno inteligenco (SLAIS), Slovensko društvo za kognitivne znanosti (DKZ) in druga slovenska nacionalna akademija, Inženirska akademija Slovenije (IAS). V imenu organizatorjev konference se zahvaljujemo združenjem in institucijam, še posebej pa udeležencem za njihove dragocene prispevke in priložnost, da z nami delijo svoje izkušnje o informacijski družbi. Zahvaljujemo se tudi recenzentom za njihovo pomoč pri recenziranju.

V letu 2018 bomo šestič podelili nagrado za življenjske dosežke v čast Donalda Michieja in Alana Turinga. Nagrado Michie-Turing za izjemen življenjski prispevek k razvoju in promociji informacijske družbe bo prejel prof. dr. Saša Divjak. Priznanje za dosežek leta bo pripadlo doc. dr. Marinki Žitnik. Že sedmič podeljujemo nagradi »informacijska limona« in »informacijska jagoda« za najbolj (ne)uspešne poteze v zvezi z informacijsko družbo. Limono letos prejme padanje državnih sredstev za raziskovalno dejavnost, jagodo pa Yaskawina tovarna robotov v Kočevju. Čestitke nagrajencem!

Mojca Ciglarič, predsednik programskega odbora

Matjaž Gams, predsednik organizacijskega odbora

FOREWORD - INFORMATION SOCIETY 2018

In its 21st year, the Information Society Multiconference (http://is.ijs.si) remains one of the leading conferences in Central Europe devoted to information society, computer science and informatics. In 2018, it is organized at various locations, with the main events taking place at the Jožef Stefan Institute.

Information society, knowledge and artificial intelligence continue to represent the central pillars of human civilization. Will the pace of progress of information society, knowledge and artificial intelligence continue, thus enabling unseen progress of human civilization, or will the progress stall and even stagnate? Will ICT and AI continue to foster human progress, or will the growth of human, demographic, social and environmental problems stall global progress? Both extremes seem to be playing out to a certain degree – we seem to be transitioning into the next civilization period, while the internal and external conflicts of the contemporary society seem to be on the rise.

The Multiconference runs in parallel sessions with 215 presentations of scientific papers at eleven conferences, many round tables, workshops and award ceremonies. Selected papers will be published in the Informatica journal, which boasts of its 42-year tradition of excellent research publishing.

The Information Society 2018 Multiconference consists of the following conferences:

- Slovenian Conference on Artificial Intelligence
- Cognitive Science
- Data Mining and Data Warehouses - SiKDD
- International Conference on High-Performance Optimization in Industry, HPOI
- AS-IT-IC Workshop
- Facing demographic challenges
- Collaboration, Software and Services in Information Society
- Workshop Electronic and Mobile Health and Smart Cities
- Education in Information Society
- 5th Student Computer Science Research Conference
- International Technology Transfer Conference (ITTC)

The Multiconference is co-organized and supported by several major research institutions and societies, among them ACM Slovenia, i.e. the Slovenian chapter of the ACM, Slovenian Artificial Intelligence Society (SLAIS), Slovenian Society for Cognitive Sciences (DKZ) and the second national engineering academy, the Slovenian Engineering Academy (IAS). On behalf of the conference organizers, we thank all the societies and institutions, and particularly all the participants for their valuable contribution and their interest in this event, and the reviewers for their thorough reviews.

For the sixth year, the award for life-long outstanding contributions will be presented in memory of Donald Michie and Alan Turing. The Michie-Turing award will be given to Prof. Saša Divjak for his life-long outstanding contribution to the development and promotion of information society in our country. In addition, an award for current achievements will be given to Assist. Prof. Marinka Žitnik. The information lemon goes to decreased national funding of research. The information strawberry is awarded to the Yaskawa robot factory in Kočevje. Congratulations!

Mojca Ciglarič, Programme Committee Chair

Matjaž Gams, Organizing Committee Chair

KONFERENČNI ODBORI
CONFERENCE COMMITTEES

International Programme Committee

Vladimir Bajic, South Africa
Heiner Benking, Germany
Se Woo Cheon, South Korea
Howie Firth, UK
Olga Fomichova, Russia
Vladimir Fomichov, Russia
Vesna Hljuz Dobric, Croatia
Alfred Inselberg, Israel
Jay Liebowitz, USA
Huan Liu, Singapore
Henz Martin, Germany
Marcin Paprzycki, USA
Karl Pribram, USA
Claude Sammut, Australia
Jiri Wiedermann, Czech Republic
Xindong Wu, USA
Yiming Ye, USA
Ning Zhong, USA
Wray Buntine, Australia
Bezalel Gavish, USA
Gal A. Kaminka, Israel
Mike Bain, Australia
Michela Milano, Italy
Derong Liu, USA
Toby Walsh, Australia

Organizing Committee

Matjaž Gams, chair
Mitja Luštrek
Lana Zemljak
Vesna Koricki
Mitja Lasič
Blaž Mahnič
Jani Bizjak
Tine Kolenik

Programme Committee

Franc Solina, co-chair
Viljan Mahnič, co-chair
Cene Bavec, co-chair
Tomaž Kalin, co-chair
Jozsef Györkös, co-chair
Tadej Bajd
Jaroslav Berce
Mojca Bernik
Marko Bohanec
Ivan Bratko
Andrej Brodnik
Dušan Caf
Saša Divjak
Tomaž Erjavec
Bogdan Filipič
Andrej Gams
Matjaž Gams
Marko Grobelnik
Nikola Guid
Marjan Heričko
Borka Jerman Blažič Džonova
Gorazd Kandus
Urban Kordeš
Marjan Krisper
Andrej Kuščer
Jadran Lenarčič
Borut Likar
Mitja Luštrek
Janez Malačič
Olga Markič
Dunja Mladenič
Franc Novak
Vladislav Rajkovič
Grega Repovš
Ivan Rozman
Niko Schlamberger
Stanko Strmčnik
Jurij Šilc
Jurij Tasič
Denis Trček
Andrej Ule
Tanja Urbančič
Boštjan Vilfan
Baldomir Zajc
Blaž Zupan
Boris Žemva
Leon Žlajpah

KAZALO / TABLE OF CONTENTS

Zbornik 21. mednarodne multikonference

INFORMACIJSKA DRUŽBA – IS 2018

Zvezek G

Proceedings of the 21st International Multiconference

INFORMATION SOCIETY – IS 2018

Volume G

Sodelovanje, programska oprema in storitve v informacijski družbi
Collaboration, Software and Services in Information Society

Uredil / Edited by

Marjan Heričko

http://is.ijs.si

9. oktober 2018 / 9 October 2018
Ljubljana, Slovenia

PREFACE

This year, the Conference "Collaboration, Software and Services in Information Society" is being organised for the eighteenth time as a part of the "Information Society" multi-conference. As in previous years, the papers from this year's proceedings address actual challenges and best practices related to the development of advanced software and information solutions as well as collaboration in general.

Information technologies and the field of Informatics have been the driving force of innovation in business, as well as in the everyday activities of individuals for several decades. Blockchain technology, Big Data, intelligent solution, reference models, open standards, interoperability and the increasing responsiveness of IS/IT experts are leading the way to the development of intelligent digital service platforms, innovative business models and new ecosystems where not only partners, but also competitors are connecting and working together. On the other hand, quality assurance remains a vital part of software and ICT-based service development and deployment. The papers in these proceedings provide a better insight and/or propose solutions to challenges related to:

- Self-Assessment of Sustainability of ICT in SMEs;
- Ontology-based knowledge sharing on BPMN graphical signs using semiotics;
- Influence of notations used for conceptual design on knowledge perception;
- Application of machine learning techniques to obtain new knowledge;
- Establishment of domain specific reference models;
- Introduction of Blockchain technology into real-life use cases;
- Architectural design proposals for ensuring scalability of Blockchain platforms;
- Application of usability questionnaires when evaluating gamification and serious games
- Visualization, analysis and comprehension of complex software systems;
- Continuous software development, integration and delivery;
- Integration of source code repositories and QA tools.

We hope that these proceedings will be beneficial for your reference and that the information in this volume will be useful for further advancements in both research and industry.

Prof. Dr. Marjan Heričko
CSS 2018 – Collaboration, Software and Services in Information Society Conference Chair

PREDGOVOR

Konferenco "Sodelovanje, programska oprema in storitve v informacijski družbi" organiziramo v sklopu multikonference Informacijska družba že osemnajstič. Kot običajno, tudi letošnji prispevki naslavljajo aktualne teme in izzive, povezane z razvojem sodobnih programskih in informacijskih rešitev ter storitev kot tudi sodelovanja v splošnem.

Informatika in informacijske tehnologije so že več desetletij gonilo inoviranja na vseh področjih poslovanja podjetij ter delovanja posameznikov. Tehnologija veriženja blokov, velepodatki, inteligentne storitve, referenčni modeli, odprti standardi in interoperabilnost ter vedno višja odzivnost informatikov vodijo k razvoju inteligentnih digitalnih storitvenih platform in inovativnih poslovnih modelov ter novih ekosistemov, kjer se povezujejo in sodelujejo ne le partnerji, temveč tudi konkurenti. Napredne informacijske tehnologije in sodobni pristopi k razvoju, vpeljavi in upravljanju omogočajo višjo stopnjo avtomatizacije in integracije doslej ločenih svetov, saj vzpostavljajo zaključeno zanko in zagotavljajo nenehne izboljšave, ki temeljijo na aktivnem sodelovanju in povratnih informacijah vseh vključenih akterjev. Ob vsem tem zagotavljanje kakovosti ostaja eden pomembnejših vidikov razvoja in vpeljave na informacijskih tehnologijah temelječih storitev.

Prispevki, zbrani v tem zborniku, omogočajo vpogled v in rešitve za izzive na področjih kot so npr.:

- samoocenitev kakovosti in zrelosti IKT podpore v malih in srednje velikih podjetjih;
- deljenje znanja o grafičnih simbolih BPMN z uporabo semiotike;
- vpliv notacije, uporabljene pri oblikovanju konceptualih modelov, na dojeti nivo pridobljenega znanja;
- uporaba tehnik strojnega učenja za ekstrakcijo znanja;
- vzpostavitev domenskih referenčnih modelov;
- vpeljava tehnologije veriženja blokov v realne primere uporabe;
- arhitekturni predlogi za rešitev razširljivosti platform tehnologije veriženja blokov;
- uporaba standardnih vprašalnikov uporabnosti pri vrednotenju učinkov vpeljave igrifikacije in resnih iger;
- vizualizacija, analiza in razumevanje kompleksnih programskih sistemov;
- neprekinjen razvoj, integracija in dobava informacijskih rešitev;
- integracija repozitorijev izvorne kode z orodji za zagotavljanje kakovosti.

Upamo, da boste v zborniku prispevkov, ki povezujejo teoretična in praktična znanja, tudi letos našli koristne informacije za svoje nadaljnje delo tako pri temeljnem kot aplikativnem raziskovanju.

prof. dr. Marjan Heričko
predsednik konference CSS 2018 – Collaboration, Software and Services in Information Society Conference

PROGRAMSKI ODBOR / PROGRAM COMITTEE

Dr. Marjan Heričko
University of Maribor, Faculty of Electrical Engineering and Computer Science

Dr. Gabriele Gianini
University of Milano, Faculty of Mathematical, Physical and Natural Sciences

Dr. Hannu Jaakkola
Tampere University of Technology Information Technology (Pori)

Dr. Mirjana Ivanović
University of Novi Sad, Faculty of Science, Department of Mathematics and Informatics

Dr. Zoltán Porkoláb
Eötvös Loránd University, Faculty of Informatics

Dr. Stephan Schlögl
MCI Management Center Innsbruck, Department of Management, Communication & IT

Dr. Zlatko Stapić
University of Zagreb, Faculty of Organization and Informatics

Dr. Vili Podgorelec
University of Maribor, Faculty of Electrical Engineering and Computer Science

Dr. Maja Pušnik
University of Maribor, Faculty of Electrical Engineering and Computer Science

Dr. Muhamed Turkanović
University of Maribor, Faculty of Electrical Engineering and Computer Science

Dr. Boštjan Šumak
University of Maribor, Faculty of Electrical Engineering and Computer Science

Dr. Aida Kamišalić Latifić
University of Maribor, Faculty of Electrical Engineering and Computer Science

Dr. Gregor Polančič
University of Maribor, Faculty of Electrical Engineering and Computer Science

Dr. Luka Pavlič
University of Maribor, Faculty of Electrical Engineering and Computer Science

Self-Assessment Tool for Evaluating Sustainability of ICT in SMEs

Jari Soini
Tampere University of Technology
P.O. Box 300
FI-28101 Pori, Finland
jari.o.soini@tut.fi

Jari Leppäniemi
Tampere University of Technology
P.O. Box 300
FI-28101 Pori, Finland
jari.leppaniemi@tut.fi

Pekka Sillberg
Tampere University of Technology
P.O. Box 300
FI-28101 Pori, Finland
pekka.sillberg@tut.fi

ABSTRACT
The ever-increasing demand for ICT may compromise global objectives for emissions reduction if the aggregate effects of ICT sustainability are not considered in the business digitalization processes. In this paper, we present a free self-assessment tool enabling small and medium sized companies to evaluate the utilized ICT in terms of sustainability. The ICT4S is a free e-service, in effect, a web-based self-assessment tool that was developed in co-operation with Swiss Green IT SIG. The assessment is currently divided into five categories of sustainability questions. The categories are strategy, procurement and recycling, practices, servers and network, and Green ICT. As the result, organizations will gain a general understanding about their state of sustainability, and practical suggestions for greater eco-friendliness and sustainability of their ICT operations.

Categories and Subject Descriptors
• **Social and professional topics~Sustainability** • Information systems~Web applications

General Terms
Measurement, Performance, Human Factors.

Keywords
Sustainability, Assessment, ICT, Metrics, Web tools, E-services.

1. INTRODUCTION
The study presented in this paper aims at contributing to the business activity digitalization of companies concerning the reduction of carbon footprint and improvement of sustainability. The paper introduces a self-assessment tool developed in the research project that allows companies to self-evaluate the sustainability of the ICT exploited in the organization. The objective is to provide companies with concrete tools and proposals for actions enabling more ecological procedures in the organization. Additionally, the knowledge gained by using the self-assessment tool allows companies to become generally more aware of the distribution of energy consumption in a modern ICT infrastructure as well as the factors affecting sustainability of ICT.

2. BACKGROUND
There is a lot of evidence for significant benefits in terms of productivity and cost savings through the exploitation of ICT in the daily business activity of organizations. However, the increasingly dependent use of ICT also brings about "invisible" effects (e.g., electricity used by database servers, cloud servers, and network routers) that may not be consciously recognized [1, 2, 3, 4]. Typically users are concerned only of the electricity consumption of their own devices. The increasing demand for ICT may, in fact, compromise the national objectives for emissions reduction if the aggregate effects of ICT un-sustainability (Figure 1) are not considered in the business digitalization processes.

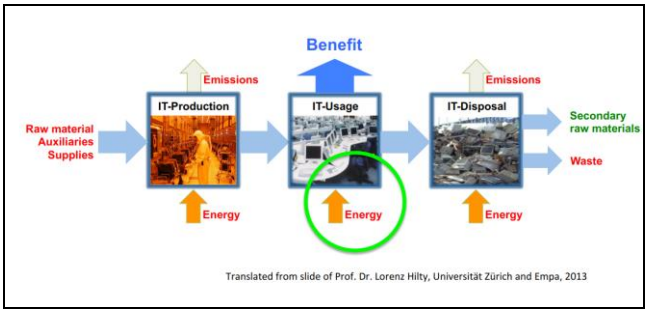


Figure 1. Environmental impacts of the ICT. [5]

In 2017, it was estimated that ICT accounted for 12% of the overall electricity consumption around the globe, and the percentage is expected to increase twice as rapidly in the future (by approximately 7% per year). Most of the energy is consumed by networks, server rooms, and computing centers, (Figure 2) the efficiency of which should urgently be improved.

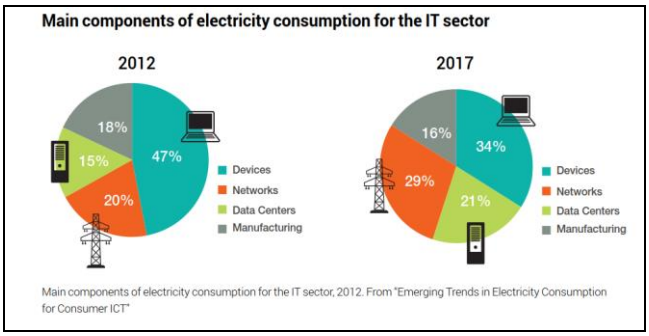


Figure 2. Electricity consumption in the ICT sector. [6]

As most of the electricity is still being generated by using fossil fuels (Figure 3), the current ICT, and its heavy usage of electrical energy, constitutes a global issue that is, unfortunately, little known outside the expert field [7, 8]. This is partly due to the users not perceiving the energy consumption of data systems operating invisibly or in the background, but rather only noticing the consumption of the terminal device, which, in reality, comprises a fraction of the overall energy consumption (Figure 2).

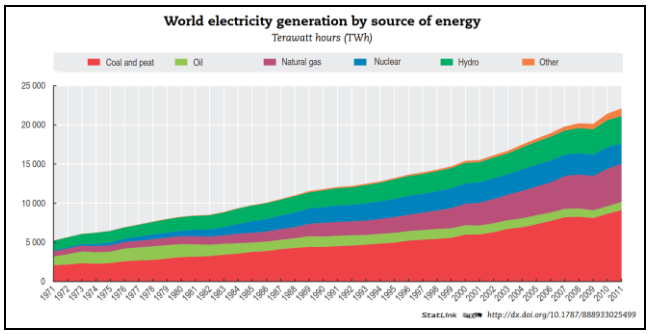
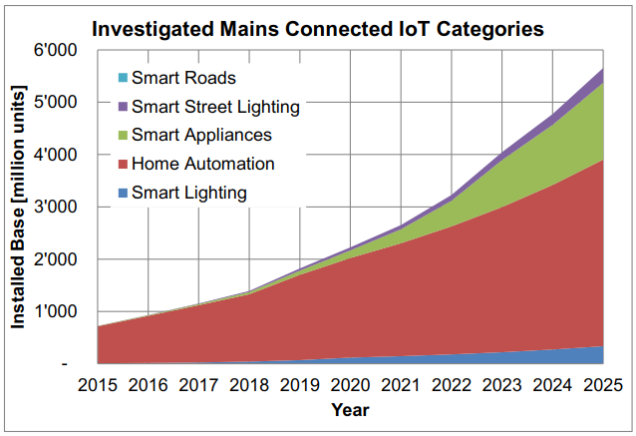
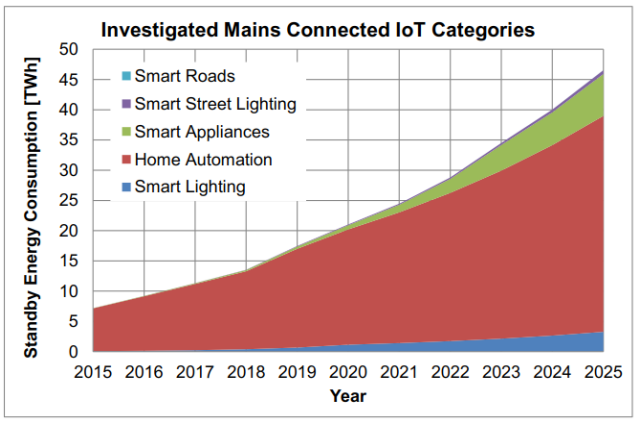


Figure 3. Electricity generation by source of energy. [9]

The problem of energy consumption due to the constantly increasing utilization of ICT is expected to further worsen (Figures 4a and 4b) through the amount of IoT devices and automatic steering systems [10]. If the majority of the predicted IoT devices and information systems supporting them are implemented by the current practices, a near-catastrophic peak demand in terms of electricity will ensue. This, in turn, will result in an increase in emissions rather than their reduction.



(a) Estimated growth.



(b) Estimated standby energy consumption.

Figures 4a and 4b. Estimated growth and impact of IoT devices. [10]

Therefore, it is essential to establish instructions and an assessment procedure to support system planning to improve sustainability of ICT, and, thus, to promote methods for a low-carbon economy.

In 2015 through 2017, the TUT Pori Department implemented a research project (AjaTar) with the aim of improving the digitalization of organizations and companies while promoting a low-carbon economy and sustainability. As part of the project, a technology enabling organizations to self-evaluate their ICT sustainability was developed, tested, and studied, aiming at increasing general awareness of the distribution of electricity consumption in a modern IT infrastructure in order for the organizations to be able to make ICT-related decisions more consciously than before.

The most notable added value of the project comprise an increase in knowhow and knowledge promoting easy and lightweight assessment of sustainability in terms of the organization's business activities and support processes, as well as a freely available tool for evaluating the sustainability of the ICT used in the organization. By making the sustainability issues visible, the objective was to change attitudes and conventions related to the utilization of ICT in organizations: indeed, during the project, several organizations distinctly declared their need to recognize practices promoting sustainable development as well as invest in an eco-friendly image.

3. ICT4S SELF-ASSESSMENT TOOL

During the last six years, the SEIntS research group from TUT Pori Department has studied, developed, and piloted innovative ICT solutions in cooperation with local organizations. Additionally, SEIntS has collaborated with, for example Keio University in Japan as well as with various information society associations, for example, in Switzerland regarding the Green IT and assessment of datacenters. As a result of the AjaTar project, an open self-assessment website for organizations to quickly and easily evaluate the ecological aspects of their ICT-related operations was published at the end of 2017. The self-assessment tool, developed in collaboration with Green IT SIG, a Swiss Green IT information special interest group, is based on the assumption that most of the ICT equipment used in an organization is controllable, enabling the relatively easy adjustment of various functions. With the assessment tool developed in the project, it is possible to increase knowledge about the ecological aspects related to the use of ICT in organizations and, thus, affect their operations and practices. Based on the self-assessment, the organization is offered overall evaluation of the current state and propositions for practices for more sustainable ICT operations.

The self-assessment tool is freely available on a dedicated website for sustainable ICT [11]. On the landing page of the tool (Figure 5) there is a welcoming message that explains the goals of the assessment. There is also information of the privacy solution that is used to guarantee all the information of the assessor's company. The privacy solution is based on the HTML5 local storage concept. The assessment menu is currently divided into five categories of sustainability questions and the information of the organization to be evaluated. The categories are: *strategy*, *procurement and recycling*, *practices*, *servers and network*, and *Green ICT*.

Figure 5. Welcoming the assessors.

Each of the categories comprises several questions and additional text that explains the current issue to the assessor. While trying to answer the questions, the assessor also receives background information on the current topic. In Figures 6 and 7, the assessor is facing questions concerning the strategy and practices at the office.

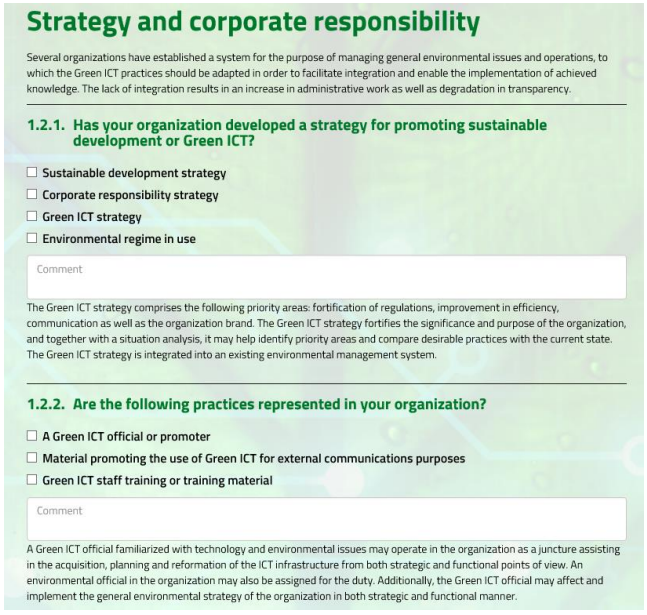


Figure 6. Assessing the strategy.

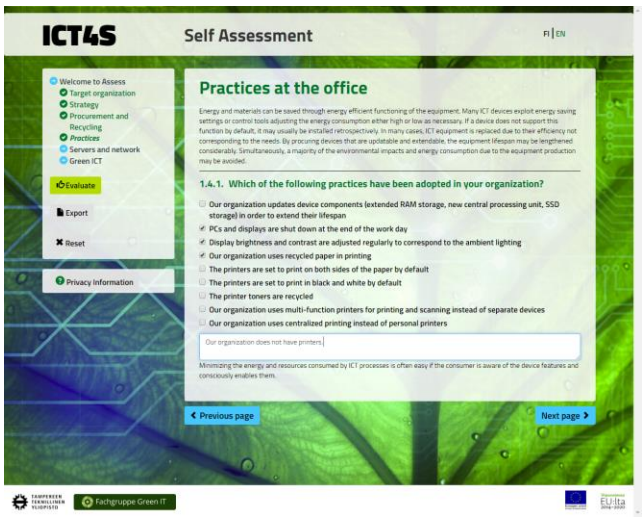


Figure 7. Assessing practices at the office category.

After assessing all categories, the assessment tool calculates and shows an evaluation of the given answers. The results are first shown in a short form as in Figure 8, but users can explore the results more carefully by selecting "Display detailed evaluation." The percentage and the color of the beams give a fast response of the maturity of the different categories. In the case of 100% and a green beam, the user can be satisfied with the sustainability state of their company in that certain category. In the case of low percentages (0 - 70%) or yellow or even red beams, the evaluation shows that there is room for improvement. In such a case, the user may find the detailed evaluation useful when planning concrete actions for these improvements.

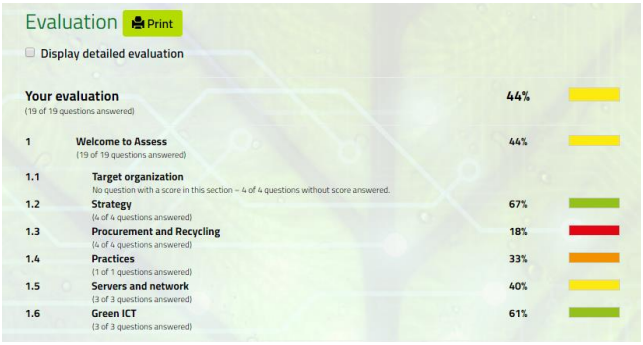


Figure 8. Brief results of the assessment.

The detailed evaluation can be shown by selecting the corresponding option in the user interface (see Figure 9). The user is also able to print the results – hopefully in a sustainable way, for example using an e-format such as Portable Document Format (PDF).

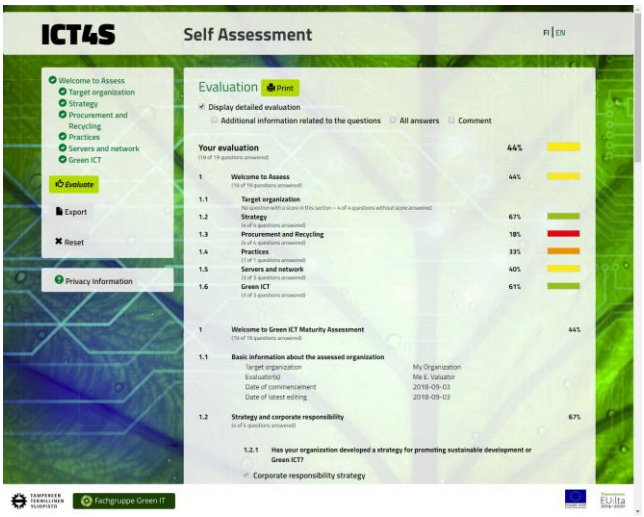


Figure 9. Detailed results of the assessment.

The assessment tool has now been in use for several months. Unfortunately, we do not have the exact statistics concerning the usage of the tool. However, we piloted the tool with the assistance of local companies before launching it last December. Since the piloting groups were satisfied with the tool and because we wanted to keep our promises regarding the privacy of the assessments, we did not implement any logging system in it.

We have planned to enhance the tool with a new capability – aiming to enable an easy way to estimate the carbon footprint of the ICT usage in a company. It will not be fully scientific life cycle assessment (LCA) but a practical version of such targeted to

non-professionals in the field of sustainability. The reasoning for this new capability is that we anticipate that by introducing easy assessment tools we will be able to raise the awareness of companies in terms of sustainability issues and thus help them to develop their business processes toward a sustainable state.

4. RESULTS AND FUTURE WORK

This paper presented the ICT4S self-assessment tool enabling companies and other organizations to evaluate the utilized ICT in terms of a low-carbon economy and sustainability and thus improve their image as well as resource efficiency. As the result, organizations will gain a general understanding of the current sustainability state of their ICT and practical suggestions for more eco-friendly and sustainable operations.

The role of the TUT Pori Unit was to function as a producer and facilitator of new knowledge. The applied project aimed at contributing to the business development with TUT Pori Unit acting as a distributor of knowledge and knowhow as well as an innovator. Within the project, the accumulation of diverse energy-related knowhow and knowledge and exploitation of sustainable solutions of ICT in organizations were successfully implemented.

Further development is planned to be realized in the ICT4LC project launched at the beginning of 2018. It focuses on examining contemporary information processing that is based on mobile and 'thin clients' as well as the increasing utilization rate of information networks and cloud computing. The new project explores tools for assessing the energy efficiency of business activities and support processes as well as planning procedures of business processes, promoting responsible and sustainable utilization of ICT in organizations.

5. ACKNOWLEDGMENTS

Our thanks to Niklaus Meyer and Beat Koch from Swiss Green IT SIG for collaboration.

6. REFERENCES

[1] Hilty, L., Arnfalk, P., Erdmann, L., Goodman, J., Lehmann, M., Wager, A.P. 2006. The relevance of information and communication technologies for environmental sustainability – A prospective simulation study. *Environmental Modelling & Software 2006*, vol. 21, issue 11, 1618-1629.

[2] Hilty, L. 2008. Information technology and sustainability: Essays on the relationship between ICT and sustainable development. *Books on Demand*, Norderstedt.

[3] Amsel, N., Ibrahim, Z., Malik, A. and Tomlinson, B. 2011. Toward sustainable software engineering: NIER track. published in *33rd International Conference on Software Engineering (ICSE)*, 21-28 May 2011, Honolulu, USA.

[4] Baliga, J., Hinton, K., Ayre, R. and Tucker, R.S. 2009. Carbon footprint of the internet. *Journal of Australia*, vol. 59, no. 1, 5.1-5.14.

[5] Hilty, L. and Aebischer, B. (eds.). 2015. *ICT Innovations for Sustainability*. Advances in Intelligent System and Computing 310, Springer International Publishing, Switzerland.

[6] Corcoran, A. and Andrae, A. 2013. Emerging Trends in Electricity Consumption for Consumer ICT. Retrieved August 22, 2018 from https://www.researchgate.net/profile/Anders_Andrae/publication/255923829_Emerging_Trends_in_Electricity_Consumption_for_Consumer_ICT/

[7] Pickavet, M., Vereecken, W., Demeyer, S., Audenaert, P., Vermeulen, B., Develder, C., Colle, D., Dhoedt, B. and Demeester, P. 2008. Worldwide energy needs for ICT: The rise of power-aware networking. In proceedings of *2nd International Symposium on Advanced Networks and Telecommunication Systems*, 1-3.

[8] Lambert, S., and Van Heddeghem, W. 2012. Worldwide electricity consumption of communication networks. *Optics Express*, vol. 20, no. 26, 513-524.

[9] OECD Factbook 2014: Economic, Environmental and Social Statistics. Retrieved August 27, 2018 from http://dx.doi.org/10.1787/888933025499

[10] International Energy Agency. 2016. Energy Efficiency of the Internet of Things, Technology and Energy Assessment Report. Prepared for IEA 4E EDNA. Retrieved August 27, 2019 from https://www.iea-4e.org/document/384/energy-efficiency-of-the-internet-of-things-technology-and-energy-assessment-report

[11] Tampere University of Technology. 2017. ICT4S Self Assessment. Retrieved August 27, 2018 from https://green-ict.fi/arviointi/?lang=en

Reference Standard Process Model for Farming to Support the Development of Applications for Farming

Rok Rupnik
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia
+386 1 479 8266
rok.rupnik@fri.uni-lj.si

ABSTRACT
The paper introduces the idea and the concepts of a Reference Standard Process Model (RSPMF) which are based on the concepts of COBIT, an IT governance framework used worldwide. Our research on RSPMF is focused in two directions. First, RSPMF is aimed at becoming a support for Product Managers in software companies developing software products or IoT systems. Namely, each process in RSPMF is described through the following components: Process goals, process metrics, KPI's (Key Performance Indicators) and process activities, Second, RSPMF is aimed to help managers or owners of bigger farms in farm management. The paper introduces research in the progress state of our research.

Categories and Subject Descriptors
D.2.2 [**Requirements/Specifications**]: *Tools.*

General Terms
Farming, Standardization, Process model.

Keywords
Standard Process Model, COBIT, Transformation of model.

1. INTRODUCTION
In recent years, farming has become an area with extensive need for the use of information systems and IoT technologies [1]. The experience gained in an EU funded project has revealed that software companies have diverse and unequal knowledge and understanding of farming processes, activities within processes and metrics. This causes a problem when software products and IoT systems need to be integrated. There are many software products and IoT systems on the market today, but each of them covers a quite narrow functional area and, for the treason the integration, is simply a necessity [2].

The Reference Standard Process Model is one way to help Product Managers at software companies in removing the gap of diverse and unequal knowledge and understanding of farming processes, activities within processes and metrics. The reference model can become a common denominator, a kind of *Esperanto*, as a knowledge base for the development of software products and IoT systems for farming. The reference model, on the other hand, will also help farm managers and owners in farm management.

We built and designed a *Reference Standard Process Model for Farming* (RSPMF) based on the idea and concepts of the COBIT framework, which is defined for the area of *IT governance* [3], [4]. This paper introduces the research in progress and the concepts we have managed to define so far: Domains, processes and elements of process description. We also introduce the current list of processes and domains.

The structure of the paper is as follows. The second chapter introduces the EU funded project AgroIT, during which the idea for the Reference Standard Process Model arose. Only aspects of the project relevant for the content of this paper are introduced. The third chapter introduces key findings from the AgroIT project which led to the idea of RSPMF. To support the idea of RSPMF the COBIT framework for IT governance is also introduced, since many concepts of RSPMF are taken from the COBIT framework. The fourth chapter introduces the RSPMF, its concepts, draft list of domains and their processes, and the methodology to facilitate the sustainability of RSPMF. The last chapter contains the conclusion and directions for future work on the RSPMF.

2. EXPERIENCE GAINED IN THE AgroIT PROJECT
AgroIT was an EU funded project covering various previously mentioned aspects and problems in today's implementation of IT and IoT in farming [5], [6]. First, the project included the implementation of ERP systems for farming: A traditional ERP system for small and medium enterprises which, additionally, also has modules for livestock, fruit growing, winery, etc. [7]. This area of farm management was covered, which was the subject of several papers in recent years [8][1], [2], [6], [7], [9], [10]. Second, the project included the implementation of a decision support system based on advanced methods to support decision processes in farming [8]. This way, the area of the use of decision support within farm management was covered [1], [6]. Third, the project included the implementation of IoT systems where various sensors were used to collect data about several measurements [2], [11], [12]. Having (a lot of) data available is the basis for farm management and operations of farms [13]. Fourth, the project also covered the implementation of the cloud integration platform. All applications and IoT systems were integrated through the cloud integration platform to facilitate data exchange between them [6], [12], [14].

Six software companies (they were called *software partners* during the project) cooperated in the AgroIT project with their software products: Applications, IoT systems and the cloud integration platform. Each software company "contributed" their product to the project and, during the project, software products were improved significantly, i.e. upgraded and extended. They were also improved implicitly through integrations between each other.

For the *pilot use* of integrated software products and IoT systems several *pilot projects* were organised in 5 EU countries by *pilot*

partners. *Pilot partners* did not do software implementation in the project, but supported pilot farms in the use of software products. For that reason, pilot partners were organisations with extensive knowledge in agriculture and experience in consulting for farming.

3. KNOWLEDGE OF FARMING FOR IMPLEMENTATION OF SOFTWARE PRODUCTS AND IoT SYSTEMS FOR FARMING

Improving software products and IoT systems was based on and extending existing functionalities of software products and IoT systems and upgrading them with new ones. The key goal of the project was to design functionalities which base on integration between software products and IoT systems. This means that a software product also can use data from another software product or IoT system.

During the analysis and design phase it has become apparent that software partners have diverse and unequal knowledge and understanding of farming processes, activities within processes and metrics. The gap was even bigger when compared to the knowledge and understanding of the *pilot partners*.

The diversity mentioned, and having the expertise of COBIT, has, step-by-step, led to the idea of transferring the idea of COBIT to be used for farming [3], [4].

3.1 COBIT framework for IT governance

COBIT has, in recent years, become a de-facto Standard for IT governance in companies and organisations. COBIT defines a set of generic processes (IT processes) for the management of IT. For each IT process the following is defined: Process inputs and outputs, goals of the process, key process activities, metrics of the process (performance measures), and levels of process maturity (maturity model) [3]. The development of COBIT has been progressing since 1996, from version 1 to the current version 5. COBIT is the result of several working groups of highly experienced experts as coordinated work owithin ISACA, which is an international professional association focused on IT governance.

COBIT is defined as a process model which divides IT into four domains: Plan and Organise, Acquire and Implement, Deliver and Support, and Monitor and Evaluate). Domains have altogether 34 defined IT processes.

The schema below shows a meta model of COBIT and all of its concepts. The schema reveals the business orientation of COBIT: The aim of defining the COBIT framework is to align IT and business where business goals dictate IT goals [3], [4].

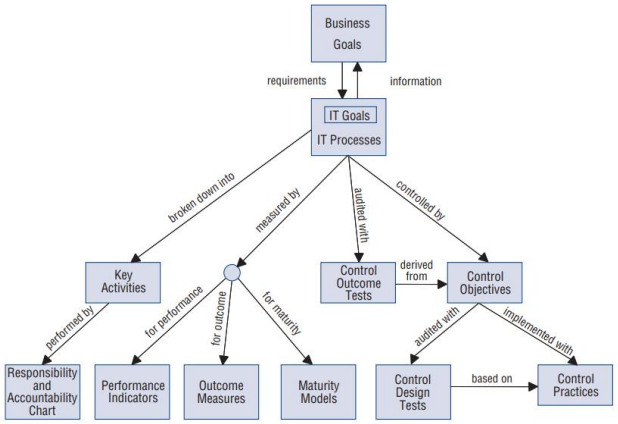


Figure 1. COBIT meta model [3]

A detailed explanation of the schema, i.e. a detailed explanation of the concepts and relations between them is beyond the scope of this paper.

3.2 The idea of the Standard Process Model for Farming

The idea and concept of the previously introduced COBIT framework and the problems based on the diversity of knowledge of partners in the project initiated the idea of a Standard Process Model for farming. COBIT is,, therefore based on various concepts, and those concepts can be used and adapted in other areas as well, not only in IT governance. The idea and concepts of COBIT were already transferred and used in the governance of Flood Management [15] and Nursing [16].

The transfer of the idea and concepts of a particular Standard or framework to another area, in this case the transfer of COBIT to the area of farming, does not mean a one-to-one transfer. Some concepts of *source area* (in this case, IT governance), might not be relevant or have any sense in the *destination area* (in this case farming). For this reason, a successful and significant transfer with useful outcome can only be achieved through:

- Good understanding of the idea and concepts of the framework of the *source area* (in this case COBIT),
- Extensive knowledge and experience on the *destination area*: Processes and their activities, metrics, responsibilities, rules, etc.

4. REFERENCE STANDARD PROCESS MODEL FOR FARMING (RSPMF)

As can be concluded based on the previous discussion, we designed RSMPF on the idea and concepts of COBIT 4.1 [3]. In the literature we so far haven't found any paper representing a Standard Process Model for Farming.

4.1 The concepts of RSMPF

Processes are divided on three hierarchical levels which are called *domains*: *Govern and Monitor* (GM), *Plan and Manage* (PM) and *Implement and Execute* (IE).

Farming has several branches: Livestock, fruit growing, agriculture, winery (viticulture), etc. RSMPF enables modular definition of processes for every branch of farming. For the *Govern and Monitor* domain only *common processes* are defined, for the other two domains, a *process module* is also added for every branch of farming. For now, only the *process module* for *livestock* is defined for domains PM and IE.

Each process is described through the following components: *Process goals*, *process metrics*, KPI's (*Key Performance Indicators*) and *process activities*.

Each process has a *unique code*, which reveals the *domain* to which the process belongs and the *process module*. The code of *Common Processes* is CP and the code for *LiveStock* is LS.

The aim of defining RSPMF is not to prevail over any existing Standard for farming. RSMPF is defined and structured to be opened and enables the reference to any existing Standard in the *process description section.*

4.2 Target groups and aimed benefits of RSPMF

When designing a Standard Process Model, regardless of the area it is intended for, the group designing it must first decide which are

the target groups who will use the model, and what should be the benefits of its use. For target groups this should become a Reference Standard Process Model.

We designed RSPMF for the following groups:

- **Product Managers** in software companies which develop software products and IoT systems for farming. As can be revealed from our discussion, we noticed the need for a Standard Process Model,
- **Managers and owners of bigger farms**: COBIT is the first place aimed at bigger companies. Each Standard Process Model should, in our opinion, be sized for bigger institutions (organisations in general). Smaller institutions then use it to the extent for which they believe is suitable for them. We followed this approach in the designing of the RSPMF.

The aimed benefits for Product Managers are as follows:

- Based on experience from the AgroIT project, we can state that there is a diversity of farming knowledge of Product Managers in software companies. RSPMF will become a common denominator, a kind of *Esperanto* as a knowledge base for the development of software products and IoT systems for farming,
- We expect the integrations between various software products and IoT systems to be more straightforward and "softer" if Product Managers will base functionalities on RSPMF.

We are designing RSPMF to reach several aimed benefits for managers and owners of bigger farms:

- Knowledge and experience of farming experts and academics will, step by step, be transferred to RSPMF. We could say that RSPMF introduces the best practices for farming,
- RSPMF provides the best practice guidelines for processes and their activities on farms. This helps managers ensure that the processes perform according to best practice,
- Metrics and KPI's are defined for processes. This helps managers to set goals and execute monitoring. This lowers various risks,
- Managers can identify gaps in process execution and monitoring. This helps them close the gaps identified and improve processes,
- Managers can be better prepared for any auditing. If a particular audited farm will be "RSPMF compliant", then this will increase the trust of auditors,
- Not only managers, but also other personnel working on farm can learn about processes, metrices and KPI's.

4.3 Draft list of domains and their processes

We already have defined a draft list of domains and their processes.

Govern and Monitor (GM):

- GM.01: Define and maintain strategy
- GM.02: Ensure profitability
- GM.03: Ensure risk governance
- GM.04: Ensure machinery and equipment governance
- GM.05: Ensure IT and innovation governance
- GM.06: Ensure compliance with legislation
- GM.07: Enable external and internal control
- GM.08: Manage and monitor process definition and change
- GM.09: Implement and monitor implementation of strategy

Plan and Manage (PM) – Common Processes (CP):

- PM.CM.01: Manage implementation of strategy and investments
- PM.CM.02: Manage budget and cost
- PM.CM.03: Manage financials
- PM.CM.04: Manage risks
- PM.CM.05: Manage human resources
- PM.CM.06: Manage buildings and security
- PM.CM.07: Manage products sales
- PM.CM.08: Manage suppliers
- PM.CM.09: Manage sub-contractors
- PM.CM.10: Manage certifications
- PM.CM.11: Manage environment and protection
- PM.CM.12: Manage energy consumption
- PM.CM.13: Manage energy production
- PM.CM.14: Manage farming machinery
- PM.CM.15: Manage equipment
- PM.CM.16: Manage IT
- PM.CM.17: Manage information system
- PM.CM.18: Manage innovations
- PM.CM.19: Manage investment projects
- PM.CM.20: Manage needs and expectations
- PM.CM.21: Manage knowledge and legislation
- PM.CM.22: Manage changes based on legislation demands
- PM.CM.25: Manage changes based on IT and innovation
- PM.CM.26: Manage assets
- PM.CM.27: Manage technical capacity
- PM.CM.28: Manage internal control

Plan and Manage (PM) – LiveStock (LS):

- PM.LS.01: Manage animal sales
- PM.LS.02: Manage animal purchases
- PM.LS.03: Manage animals` health and veterinary service
- PM.LS.04: Manage animal welfare
- PM.LS.05: Manage hygiene
- PM.LS.06: Manage animal feeding and grazing
- PM.LS.07: Manage animal reproduction
- PM.LS.08: Manage animal breeding plan

Implement and Execute (IE) – Common Processes (CP):

- IE.CM.01: Perform internal control
- IE.CM.02: Perform farm accounting
- IE.CM.03: Perform maintenance of buildings
- IE.CM.04: Perform employments and other Human Resource issues
- IE.CM.05: Perform product sales
- IE.CM.06: Perform purchases of equipment
- IE.CM.07: Perform purchases of farming machinery
- IE.CM.08: Perform purchases and implementation of software products
- IE.CM.09: Perform asset maintenance
- IE.CM.10: Perform purchases

Implement and Execute (IE) – LiveStock (LS):

- IE.LS.01: Perform animal feeding
- IE.LS.02: Perform animal movements and grazing
- IE.LS.03: Preform animal health checking and health treatment
- IE.LS.04: Perform sales of animals

- IE.LS.05: Perform purchasing of animals
- IE.LS.06: Perform animal selection
- IE.LS.07: Perform animal reproduction

4.4 Concepts of methodology to facilitate the sustainability of RSPMF

COBIT was first issued in 1996, and this means that it has been going through evolution, where experts from the whole world participated. COBIT is now version 5, but had several versions before that [3], [4].

To facilitate the sustainability of RSPMF, we plan a similar approach. We have plan to issue the first version in a year or year and a half. The first version will cover only livestock. We will form an international panel of experts of various profiles: Consultants, academics, Product Managers, farmers and government officials.

5. CONCLUSION AND FUTURE WORK

We have introduced the research in progress for the idea and concepts of the Reference Standard Process Model for Farming. Our aim of the design of reference model is to improve the support for managers and owners of bigger farms in farm management. Another aim is to facilitate Product Managers in development of software products and IoT systems.

In midterm, we also want RSPMF to be suitable for government and EU officials who are responsible for farming. At the moment, we plan to add the concept of maturity levels of a process. The maturity level of a process will show or indicate the level of detail and expertise with which a farm executes a process. This way, the comparison of different farms will also be possible.

We are aware that there are two phases of defining RSPMF: First, to define its concepts and structure; second, to put content in the structure of processes` descriptions. Those two phases overlap, because, while inserting the content, for sure some ideas to change structure will appear. The definition of concepts and the structure is our research mission for the next 12 months, that is how we plan it.

6. REFERENCES

[1] A. Kaloxylos *et al.*, "A cloud-based Farm Management System: Architecture and implementation," *Comput. Electron. Agric.*, vol. 100, pp. 168–179, Jan. 2014.

[2] S. Fountas *et al.*, "Farm management information systems: Current situation and future perspectives," *Comput. Electron. Agric.*, vol. 115, pp. 40–50, 2015.

[3] ISACA, *COBIT 4.1*. 2007.

[4] ISACA, *COBIT5: Enabling Processes*. 2012.

[5] L. Ruiz-Garcia and L. Lunadei, "The role of RFID in agriculture: Applications, limitations and challenges," *Comput. Electron. Agric.*, vol. 79, no. 1, pp. 42–50, Oct. 2011.

[6] A. Kaloxylos *et al.*, "Farm management systems and the Future Internet era," *Comput. Electron. Agric.*, vol. 89, no. null, pp. 130–144, Nov. 2012.

[7] C. N. Verdouw, R. M. Robbemond, and J. Wolfert, "ERP in agriculture: Lessons learned from Dutch horticulture," *Comput. Electron. Agric.*, vol. 114, pp. 125–133, 2015.

[8] R. Rupnik, M. Kukar, P. Vračar, D. Košir, D. Pevec, and Z. Bosnić, "AgroDSS: A decision support system for agriculture and farming," *Comput. Electron. Agric.*, no. November 2017, 2018.

[9] R. Nikkilä, I. Seilonen, and K. Koskinen, "Software architecture for farm management information systems in precision agriculture," *Comput. Electron. Agric.*, vol. 70, no. 2, pp. 328–336, Mar. 2010.

[10] C. G. Sørensen *et al.*, "Conceptual model of a future farm management information system," *Comput. Electron. Agric.*, vol. 72, no. 1, pp. 37–47, Jun. 2010.

[11] J. De Baerdemaeker, *Precision Agriculture Technology and Robotics for Good Agricultural Practices*, vol. 46, no. 4. IFAC, 2013.

[12] J. Santa, M. A. Zamora-Izquierdo, A. J. Jara, and A. F. Gómez-Skarmeta, "Telematic platform for integral management of agricultural/perishable goods in terrestrial logistics," *Comput. Electron. Agric.*, vol. 80, no. null, pp. 31–40, Jan. 2012.

[13] J. W. Jones *et al.*, "Toward a new generation of agricultural system data, models, and knowledge products: State of agricultural systems science," *Agric. Syst.*, vol. 155, pp. 269–288, 2017.

[14] J. W. Kruize, R. M. Robbemond, H. Scholten, J. Wolfert, and a. J. M. Beulens, "Improving arable farm enterprise integration - Review of existing technologies and practices from a farmer's perspective," *Comput. Electron. Agric.*, vol. 96, pp. 75–89, 2013.

[15] M. Othman, M. Nazir Ahmad, A. Suliman, N. Habibah Arshad, and S. Maidin, "COBIT principles to govern flood management," *Int. J. Disaster Risk Reduct.*, vol. 9, 2014.

[16] M. Burnik, "The Approach for the Presentation of Nursing Processes," University of Primorska, 2011.

Semiotics of graphical signs in BPMN

Saša Kuhar
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
sasa.kuhar@um.si

Gregor Polančič
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
gregor.polancic@um.si

ABSTRACT
The terminology of graphical signs (e.g. icons, symbols, pictograms, markers etc.) is ambiguous in academic articles. This is the same with articles focusing on graphics in business notations, although concepts of graphical elements in notations are well defined. In semiotics, on the other hand, the concepts related to signs are defined in detail. In this paper, we examined linguistic terms that are used for describing graphical elements in BPMN specifications (BPMN being the de-facto Standard of business notations), and related them to the terminology specified in semiotics. We created a Sign ontology with BPMN graphical signs as ontology instances. The ontology can be used by researchers to share common knowledge about concepts of signs, symbols, icons, and indices, as well as the knowledge on BPMN graphical signs.

Categories and Subject Descriptors
H.1.m [**Information Systems**]: Models and Principles, Miscellaneous.

General Terms
Management, Documentation, Design, Languages, Theory.

Keywords
Business Process Model and Notation, BPMN, Semiotics, Ontologies, Graphical signs, icons.

1. INTRODUCTION
Business process diagrams provide a graphical notation for specifying business processes. Among many business notations, Business Process Model and Notation (BPMN) is known as the de-facto Standard [1]. BPMN consists of execution semantics and notation, the latter including graphical elements such as shapes, arrows, icons, and labels. Those elements are all signs where each has a defined meaning and represents a certain concept.

However, terminology for graphical elements (e.g. icon, sign, or shape) is not used consistently among researchers in this domain. If one, for example, wants to perform a literature search on icons in BPMN, the term *icon* does not incorporate all linguistic terms that different authors use in their articles (other words for *icon* can be *pictogram*, *symbol*, *sign*, *marker* etc). Even in BPMN specifications [2], those terms are not used uniquely, but with loosely defined synonyms.

With this situation in mind we formulated the following research questions:

RQ1: What are the linguistic terms that are used in the BPMN specification for graphical shapes, graphical icons, and other visual signs?

RQ2: Can we categorize graphical signs from BPMN according to semiotic studies?

We organized the remainder of the article as follows. The next chapter presents the theoretical background. Chapters 3 and 4 represent the main objective of this paper – answering the research questions. The conclusion is given in the last chapter.

2. BACKGROUND

2.1 Semiotics
Semiotics is the study of signs and symbols (not only visual) and their use or interpretation. For the purpose of the terminology definition, we will sum the book of Daniel Chandler *Semiotics: The basics* [3], which offers a comprehensive explanation of the field, including many views of modern theoreticians. There are two main traditions in contemporary semiotics: From Ferdinand de Saussure and Charles Sanders Peirce.

Saussure's model of signs consists of two parts: **Signifier** (the form that the sign takes) and **signified** (the concept to which it refers). The **sign** is then the whole that results from the association of the signifier and the signified (Figure 1 on the left). For Saussure, both signifier and signified take non-material form rather than substance. Nowadays, common adoption of his model takes a more materialistic form, where the signifier is commonly interpreted as the material that can be seen, heard, touched, smelled or tasted. Being concerned mostly with linguistics, Saussure stressed that the relationship between the signifier and the signified is relatively arbitrary: There is no inherent, essential, transparent, self-evident or natural connection between the signifier and the signified – between the sound of a word and the concept to which it refers [3].

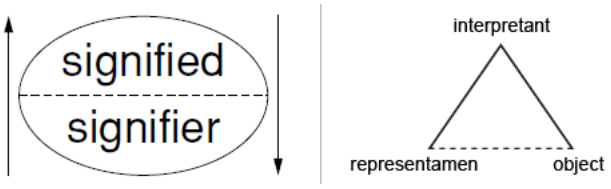


Figure 1: Saussure's model of signs on the left and Peirce`s model of signs on the right

Peirce, on the other hand, introduced a three-part model consisting of: **Representamen** (the form which the sign takes, also called "*sign vehicle*" or, in the Saussurean model, the *signifier*), **interpretant** (the sense made of the sign, or *signified* in Saussure's model). and **object** (something beyond the sign to which it refers, also called the *referent*). In this model, the **sign** is the unity of what is represented (the *object*), how it is represented (the *representamen*) and how it is interpreted (the *interpretant*) (Figure 1 on the right). The term sign is often used loosely and

confused with signifier or representamen. However, the signifier or representamen is the form in which the sign appears, whereas the sign is the whole meaningful unity [3].

2.1.1 Symbol, Index, Icon

In addition to his sign model, Peirce offered a classification of signs, based on the relationship between representamen and its object or its interpretant, or, in Sausurres' terms, the *relationship between signifier* and *signified*. Dependent upon the relationship being more arbitrary, directly connected, or more resembling, three types of signs are possible: **Symbol**, **index,** and **icon** respectively.

SYMBOL represents a relationship where the signifier does not resemble the signified, but which is *arbitrary* or conventional. The relationship must be agreed upon and learned, such as in language (letters, words, phrases, and sentences), numbers, Morse code, traffic lights or national flags.

INDEX denotes a relationship where the signifier is not arbitrary, but *connected directly* (physically or causally) to the signified, which can be observed or inferred. An index *indicates* something (that is, necessarily, existent). Examples are natural signs (smoke, thunder, footprints), medical symptoms (pain, a rash, pulse-rate), measuring instruments (thermometer, clock), 'signals' (a knock on a door, a phone ringing), recordings (a photograph, a film, video shot), personal 'trademarks' (handwriting, catchphrases).

ICON represents a relationship where the signifier is perceived as *resembling* or *imitating* the signified – being *similar* in possessing some of its qualities, like a portrait, a cartoon, a scale-model, onomatopoeia, metaphors, sound effects in radio drama, a dubbed film soundtrack and imitative gestures. [3]

2.1.2 Synonyms of terms

The terminology from semiotics is used rarely in popular language. The term *sign* in semiotics is frequently replaced by the term *symbol* in popular usage [3]. Also, several meanings of the term *icon* can be found in everyday language: a) To be iconic means that something or someone is recognized as famous, b) In computing, an icon is a small image intended to signify a particular function to the user (in semiotic terms these are *signs* which may be iconic, symbolic or indexical), c) Religious icons represent sacred, holy images [3]. If not stated otherwise, we will continue to use terms as defined in semiotics throughout this paper.

2.2 Ontologies

Ontologies are explicit formal specifications of the terms in a domain and the relationships among them [4]. They define common vocabulary and can, among other things, be used by researchers, who need to understand and share the structure of information in a domain [5]. Because of these reasons, we find them appropriate for terminology clarification in the domain of Graphical Signs in BPMN. Our research purpose is mainly definition of terms, so our ontology will, according to Obrst [6], be of the weak to moderately strong semantics, not intended to be used for machine processing or machine interpretation (at least not at this stage of our research).

3. LINGUISTIC TERMS IN BPMN SPECIFICATION

To answer the first RQ (What are the linguistic terms that are used in the BPMN specification for graphical shapes, graphical icons, and other visual signs?) we examined the BPMN specifications and mapped the specifications' terms to semiotics' terms. In BPMN specifications the signs are denominated as follows: The term BPMN element represents the term signified, the terms shape, object, marker, indication, icon and depiction stand for signifier. The answer to RQ1 and a detailed meaning of each BPMN term is provided in Table 1.

Table 1: Linguistic terms used in BPMN specifications

Semiotics' terms	BPMN terms	Detailed meanings in BPMN specification
Signified	BPMN element	Concepts in business notation
Signifier	Shape	Graphical element
	Object	Basic shape (e.g. circle representing simple event)
	Marker, Indicator or Icon	Graphical icon that can be included in an object (e.g. message icon)
	Depiction	Graphical example of the usage

As we can observe from the Table above, many linguistic terms are used for *signifier*, some of which are not used consistently (e.g. marker, indicator, and icon). The only term from semiotics that is used in BPMN specifications is the term *icon*, that is used to denote a graphical icon and stands for the term *signifier*.

4. ONTOLOGY CONSTRUCTION

For the purpose of Ontology construction and answering RQ2 (Can we categorize graphical signs from BPMN according to semiotic studies?), we followed recommendations in *Ontology Development 101: A Guide to Creating Your First Ontology* [5]. The authors suggest taking the following 7 steps for ontology creation: Step 1. Determine the domain and scope of the ontology, Step 2. Consider reusing existing ontologies, Step 3. Enumerate important terms in the ontology, Step 4. Define the classes and the class hierarchy, Step 5. Define the properties of classes, Step 6. Define the facets of the slots, and Step 7. Create instances. Steps 4 and 5 are closely intertwined and can be executed simultaneously.

4.1 Domain and scope of BPMN Sign ontology

For the domain definition, the authors [5] propose answering several questions. Our answers are provided below, after the proposed questions.

What is the domain that the BPMN Sign ontology will cover?
Signs in BPMN

What are we are going to use the ontology for?
To share a common understanding of knowledge about signs among researchers, and to be able to reuse and analyze domain knowledge.

For what types of questions should the information in the ontology provide answers?
Definitions of concepts in semiotics and relationships among them, categorization of BPMN graphical signs according to semiotics' concepts, and the frequency of occurrence of sign types in BPMN.

Who will use and maintain the ontology?
The ontology will be maintained and used by us and will be available for other interested researchers.

To determine the scope of the ontology, a list of **competency questions** can be used that ontology will be able to answer [5].

The competency questions we defined are listed next.

- What does the term icon mean?
- How do icons, indices, and symbols correlate?
- Which type of sign (icon, index or symbol) is used most in BPMN?
- Are symbols always arbitrary, or can they convey a certain degree of meaning?

4.2 Reuse of existent ontology

With a literature search we found no existing ontologies in the domain of signs or icons. However, we identified a Business Process Modelling Ontology (BPMO) that has been built automatically, starting from the XML schemas contained in the BPMN 2.0 specifications from OMG [7]. It contains all the BPMN elements and their relationships as defined BPMN specifications. The class that is related most closely to our research domain (Graphical Signs) is *DiagramElement* and its subclasses (Figure 2). This class is, in BPMN specifications, defined under BPMN Diagram Interchange (BPMN DI) meta-model and schema for the purpose of the unambiguous rendering of BPMN diagrams in different tools [2].

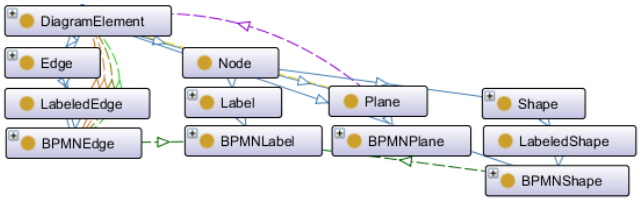


Figure 2: *DiagramElement* class and its subclasses in BPMO, visualized by the OntoGraf plugin for Protégé

As our focus in Sign Ontology is mainly on graphical signs that are, as such, not contained in BPMO, we will start our own ontology and, later, consider the options of merging both ontologies.

4.3 Definition of concepts in Sign Ontology

The next step in ontology creation is the enumeration of important terms. We defined the concepts for BPMN sign ontology from semiotics (*Sign*, *Icon*, *Index*, and *Symbol*), and from BPMN (*BasicShape*, *Activity*, *Event*, *Gateway*, and *Data*).

4.4 Relationships among concepts

For the definition of a hierarchy of classes and their properties, we will next define the relationships among three types of signs, again from semiotics.

At first sight, the relationship among the *signifier* and the *signified* (and, consequently, the types of signs) seems unambiguous, but that is not always the case. We should keep in mind that signs denote concepts (not material objects), and each person has their own understanding of a certain concept in his or her mind. Concepts cannot be represented precisely [8] therefore *icons*, for example, cannot be denoted simply as *similar*. They are defined by *perceived similarity* [3]. Also, as stated in [9], the process of sign-making is the process of the constitution of metaphor, and, therefore, *symbols* are never only *arbitrary*. Within each type, signs vary in their degree of conventionality. Therefore, we must not speak of types of signs but of modes of relationships where the difference between signs lays in the hierarchy of their properties rather than in the properties themselves [3].

Also, over time, a mode can change. Originally signs were in part iconic, in part indexical (primitive writing), and symbols come into being by development out of other signs, particularly from icons [3].

4.5 Sign Ontology construction

With the utilization of the Protégé 5.2.0 software tool and according to semiotic concepts and their relationships, we created simple Sign Ontology as follows. We created a class *Sign* (with disjoint subclasses *Icon*, *Index* and *Symbol*), a class *Relationship* (with subclasses *PrimaryRelationship* and *SecondaryRelationship*), and a class *BPMNElement* (with subclasses *BasicShape*, *Activity*, *Event*, *Gateway*, and *Data)*. We also created 2 object properties: *hasRelationshipType* (with subproperties *hasPrimaryRelationshipType* and *hasSecondaryRelationshipType*), and its inverse property *definesModeOf* (with subproperties *definesPrimaryModeOf* and *definesSecondaryModeOf*). The range of *hasPrimaryRelationshipType* is the class *Sign*, and the domain is the class *PrimaryRelationship*. We then defined 3 instances, *Arbitrary*, *Indicative* and *Similar*, and included them in the classes *PrimaryRelationship* and *SecondaryRelationship*. Next, we defined that, if a *Sign* has a *hasPrimaryRelationshipType* property of value *Similar*, it is included in the class *Icon*. Similarly, we defined classes *Index* (with *hasPrimaryRelationshipType* property value *Indicative*) and *Symbol (*with *hasPrimaryRelationshipType* property value *Arbitrary)*.

4.6 BPMN graphical shapes as Instances in Sign Ontology

To decide whether graphical signs in BPMN are of the mode icon, index or symbol, we invited 5 BPMN experts to evaluate BPMN signs and define one sign mode for each. We chose BPMN experts as they are fully familiar with the concepts (signifieds) in BPMN. Before the evaluation, the experts were acquainted with concepts from semiotics. The results of the evaluation are given in Table 2.

On six shapes, the experts agreed on the sign mode, thus defining the primary relationship between signifier and signified. For other shapes, where experts had different opinions, the mode was defined with the primary and the secondary relationship. The mode that was defined most often by experts was set for the primary relationship, and the mode that ranked second in choices was set for the secondary relationship.

As we can observe from Table 2, the majority of the signs were specified as symbols (the primary relationship is arbitrary). 6 symbols were also the only signs where experts agreed fully on the sign mode. Furthermore, in all but one symbols, the secondary mode was set as an index, and, similarly, the other way around; in all indices, the secondary mode was set as a symbol. The consensus on the primary relationship was not possible for two signs (Script task and Data object), and on the secondary relationship for one sign (Manual task). Thus, for the Script task and the Data object, the primary relationship was not set, but two secondary relationships were set. For Manual task only the primary relationship was set.

After the modes of signs were defined we included the signs into Sign Ontology. The ontology, including the instances, is shown in Figure 3. The figure represents classes as circles and relationships as lines connecting the circles. The size of the circle corresponds to the number of instances included in the class.

Table 2: Modes of BPMN signs

*	Signifier	Signified	Secondary relationship
Primary relationship: Arbitrary (Symbol)			
5		Activity	
		Gateway	
		Signal event	
		Multiple event	
		Ad-hoc sub-process	
		Complex gateway	
4		Event	Indicative (index)
		Parallel event	Indicative (index)
		Escalation event	Indicative (index)
		Link event	Indicative (index)
		Service task	Indicative (index)
		Inclusive gateway	Indicative (index)
		Parallel gateway	Indicative (index)
3		Error event	Indicative (index)
		Send task	Indicative (index)
		Receive task	Indicative (index)
		Business rule task	Indicative (index)
		Sub-process	Indicative (index)
		Exclusive gateway	Indicative (index)
		Data object collection	Similar (icon)
Primary relationship: Indicative (Index)			
4		Conditional event	Arbitrary (symbol)
		Flow	Arbitrary (symbol)
		Cancel event	Arbitrary (symbol)
		Data store	Arbitrary (symbol)
3		Compensation event	Arbitrary (symbol)
Primary relationship: Similar (Icon)			
4		Message event	Indicative (index)
3		Timer event	Indicative (index)
		User task	Arbitrary (symbol)
		Manual task	Not set
Primary relationship: Not set			
		Script task	Similar/indicative (2*)
		Data object	Similar/arbitrary (2*)

* - The number of experts who decided on this primary mode

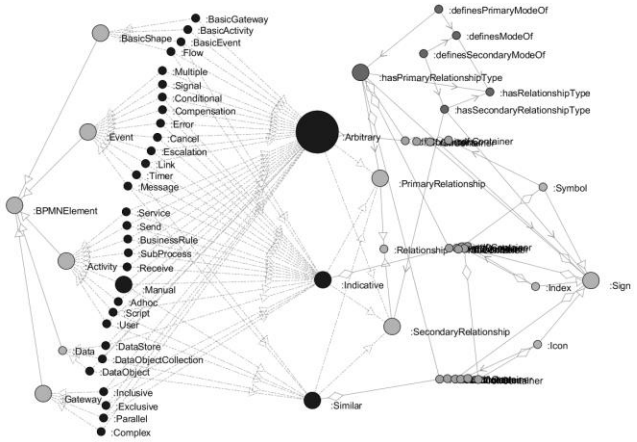


Figure 3: Sign Ontology with BPMN shapes rendered in the NavigOwl plugin for ProtégéCONCLUSION

In this paper, we mapped the linguistic terms from semiotics to linguistic terms regarding signs in BPMN specifications. We found that, in BPMN specifications, many terms are used for the term signifier, some of which inconsistently.

To correlate concepts from semiotics to BPMN graphical signs, we developed the BPMN Sign Ontology based on definitions from semiotics. We categorized each BPMN graphical sign in a mode that represents the relationship between signifier and signified. The majority of the BPMN signs are of mode symbol, following by mode index. As the meaning of symbols needs to be learned, this indicates a possible correlation with the principle of Semantic transparency from [10]. Addressing this issue, we will, in future work, examine our results further with those from [11] and other related articles.

Since the current study included only 5 experts in BPMN, resulting in possible bias, empirical research with more users is planned, as well as a thorough literature search. At this point, the BPMN Sign Ontology can, in the BPMN domain, serve for unambiguous knowledge definition and sharing.

5. REFERENCES

[1] M. Kocbek, G. Jošt, M. Heričko, and G. Polančič, "Business process model and notation: The current state of affairs," *Comput. Sci. Inf. Syst.*, vol. 12, no. 2, pp. 509–539, 2015.

[2] O.M.G., "Business Process Modeling Notation." 2011.

[3] D. Chandler and E. W. B. Hess-Lüttich, *Semiotics the Basics*, Second Edi., vol. 35, no. 6. London: Routledge, 2007.

[4] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, Jun. 1993.

[5] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," *Standford Knowl. Syst. Lab. Tech. Rep.*, pp. 1–25, 2001.

[6] L. Obrst, H. Liu, R. Wray, and L. Wilson, "Ontologies for semantically interoperable electronic commerce," *IFIP Adv. Inf. Commun. Technol.*, vol. 108, pp. 325–333, 2003.

[7] L. Cabral, B. Norton, J. Domingue, L. C. Kmi, B. Norton, and J. Domingue, "The business process modelling ontology," *Proc. 4th Int. Work. Semant. Bus. Process Manag.*, pp. 9–16, 2009.

[8] A. Fenk, "Symbols and icons in diagrammatic representation," *Pragmat. Cogn.*, vol. 6, no. 1–2, pp. 301–334, 1998.

[9] G. R. Kress and T. van Leeuwen, *Reading Images (The Grammar of Visual Design)*. London: Routledge, 1996.

[10] D. Moody, "The physics of notations: Toward a scientific basis for constructing visual notations in software engineering," *IEEE Trans. Softw. Eng.*, vol. 35, no. 6, pp. 756–779, 2009.

[11] N. Genon, P. Heymans, and D. Amyot, "Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation," in *Journal of Visual Languages & Computing*, vol. 22, no. 6, 2011, pp. 377–396.

Knowledge Perception influenced by Notation Used for Conceptual Database Design

Aida Kamišalić
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
aida.kamisalic@um.si

Muhamed Turkanović
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
muhamed.turkanovic@um.si

Marjan Heričko
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
marjan.hericko@um.si

Tatjana Welzer
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
tatjana.welzer@um.si

ABSTRACT
The paper presents an experimental study which examined the influence of the notation used for conceptual design on students' knowledge perception at higher educational study level. The results demonstrate that students' knowledge perception is higher than actual knowledge throughout the entire learning process and is correlated with the used notation.

Categories and Subject Descriptors
H.2.1 [**Database Management**]: Logical Design; K.3.2 [**Computers and Education**]: Computer and Information Science Education

General Terms
Theory, Experimentation

Keywords
entity relationship models, conceptual design, database design learning, Barker, Bachman, knowledge perception

1. INTRODUCTION
The relational databases are fundamental part of any information system. Conceptual and logical design represent important segment of almost every application. Therefore, different issues related to teaching approaches of database fundamentals and design must be adequately addressed. The introduction to databases course is one of the fundamentals of computer science and/or informatics higher education programs. It is mostly a single semester course that covers data requirements elicitation, conceptual database design, normalization, logical database design, and physical database design [3, 4, 5]. There is much research addressing issues related to teaching computer science and informatics disciplines including various aspects of databases [3, 5, 9], some research has dealt with the effectiveness of teaching approaches to database design (conceptual and logical modeling) [1, 2, 7, 8]. However, to the best of our knowledge there are no researches that dealt with knowledge perception within the databases learning environment. In order to examine the effectiveness of learning database fundamentals, depending on the notation used for conceptual design, we set-up a multi-level experimental study [7]. Different experimental instruments to evaluate the effectiveness of a teaching approach using Barker or Bachman notation for conceptual database design were developed. In contrast to Barker notation, Bachman notation incorporates elements of logical design (i.e. foreign keys) in the conceptual design level. Students' achievements were examined with regard to influencing factors throughout the learning process. Results indicated that introducing the Bachman notation and a manual transformation from a conceptual into a logical data model increased students' understanding of conceptual, logical and relational data model concepts (CLR concepts). Here we present another aspect of this study. The influence of notation used for conceptual design on student knowledge perception is examined. Research questions that are addressed and answered in the paper are (RQ1) How does the notation used for conceptual design influence students' knowledge perception? and (RQ2) Does the correlation between student's knowledge perception and actual knowledge about CLR concepts change throughout the learning process?
The structure of the paper is as follows. In Section 2, a methodological framework and experimental setting are provided. The main contribution of the paper is presented in Section 3 where results and discussion are detailed. Finally, the conclusions are presented in Section 4.

2. METHODOLOGY

2.1 Experimental framework
The study was carried out during the academic year 2016/2017 at the Faculty of Electrical Engineering and Computer Science at the University of Maribor. The experiment was performed within the Database I course. It is a single semester course that includes 45 hours of theory/practice lectures and

30 hours of laboratory work in the form of computer exercises.

The focus of the experiment was on the evaluation of students' laboratory work. Students were randomly split into two approximately equal size groups. Both groups worked on the same database modeling tasks, using the Oracle SQL Developer Data Modeler design tool. One of the groups used Bachman notation which explicitly includes the foreign key in the E-R diagram, while another group used the Barker notation, which does not explicitly include the foreign key in the E-R diagram [6].

2.2 Experimental instruments

In this section, a detailed presentation of the experimental instruments used during the study is given. The questionnaire was conducted twice: Intro-Questionnaire and Final-Questionnaire. The participation was optional in both occurrences. The questionnaire used in the study is available on the web (http://bit.ly/2wMvrVQ).

The questionnaire is split into three parts. The first part consists of mainly closed-ended questions related to basic demographic information and database design tools (Questions 1 - 6). The second part consists of a Likert scale-like multi-level table (Question 8), where participants have to cross one of the multi-level options for five basic database terms and concepts: Entity, Relationship, Attribute, Primary Key (hereinafter PK) and Foreign Key (hereinafter FK). The values of the Likert scale are as follows: (1) - I am not familiar with the term, (2) - I am familiar with the term, but not with the meaning, (3) - Undefined, (4) - I am familiar with the meaning but I do not know how to use it and (5) - I am familiar with the meaning and I know how to use it. The third part included open-ended questions, given in the form of a short test (Question 9). The short test consists of three consecutively simple tasks, whereby each is related to the previous and each presents an increase in difficulty. In order to solve the test correctly, the participants have to use a form of one-to-many (hereinafter 1:N) and many-to-many (hereinafter M:N) relationship. The participants should not be given any instructions on how to solve the test. They should be left to use any means and techniques that seem appropriate. The foreseen time limit is 20 minutes.

The purpose of the questionnaire was to examine if there was any correlation between the participant's perception of knowledge of CLR concepts (Question 8) and their actual knowledge (score on the test questions 9a, 9b, 9c). When the questionnaire was handed out the second time during the experiment, an additional closed-ended question was added to the first part (Question 7), whereby students were asked which notation they used during the laboratory work. The purpose of this particular question was to examine if there was any correlation between the notation used during the laboratory work and their knowledge (score on the test questions 9a, 9b, 9c).

In order to evaluate the questionnaire a scoring structure for the third part is needed (Question 9). The test consists of three consecutive tasks (9a, 9b, 9c), whereby each relates to the previous and each constitutes an increase in difficulty. In order to solve the first task (9a) correctly, the participants have to model an entity (i.e. person) and give it some attributes and possibly a primary key. For the second task (9b), the participants have to model an additional entity (i.e. phones) and present an 1:N relationship between the previous entity and the newly added one. For the third task (9c), the participants had to add a third entity (i.e. address), and correctly use a form of M:N relationship between the previous entities and the newly added one. In order to be able to analyze the results, five concepts are evaluated: entity, relationship, attribute, PK and FK. The scoring is as follows: if they used any possible form of the concept in their solution and if the presented use of the concept was correct, participants got a point for the concept. Thus, five points could be scored in total.

3. RESULTS AND DISCUSSION

In the next sections we report on the results achieved in the experiment. Statistical analyses were performed using IBM SPSS Statistics version 23.

3.1 Knowledge perception

An analysis was performed on related samples of the perception score and test score. It was based on data gathered from the Intro-Questionnaire and Final-Questionnaire. The data for each questionnaire was analyzed separately.

In the analysis we excluded all those records where students rated one of the concepts as undefined, thus the total number of records taken into account were 116. Therefore, we got four levels of knowledge and five different concepts. As mentioned in the previous section, part of the questionnaires was a short test. We will refer to the total test score as the test score. In order to effectively compare the actual knowledge with the perception, we normalized the results of knowledge perception so that the total score (max. 20 points) was divided by five. We will refer to the normalized perception results as the perception score. Table 1 reports on the results of the analysis which was performed using a Wilcoxon signed-rank test for related samples.

Table 1: Correlation of results for perception score and test score.

Experimental instrument	Related Samples	Asymp. Sig. (2-tailed)	N	Decision
Intro-Questionnaire	Percep. score - Test score	**0.000****	107	**Reject the null hypothesis**
Final-Questionnaire	Percep. score - Test score	**0.000****	116	**Reject the null hypothesis**

**Significant at 1%

We used the Wilcoxon signed-rank test in order to compare two not normally distributed sets of scores, one actual score and another normalized perception score, that came from the same participants, since each participant had to solve tasks and evaluate their knowledge on the aforementioned CLR topics. The Shapiro-Wilk test of normality indicated that data significantly deviates from a normal distribution (p-value below 0.05). The Wilcoxon signed-rank test returns an asymptotic significance lower than 0.01, thus rejecting the null hypothesis for related samples. The null hypothesis

states that the median of difference between the perception score and the test score will equal zero. There is a statistically significant difference between the perception score and the test score, suggesting that students' perception of their knowledge is not in accordance with their actual knowledge on CLR concepts. Figures 1 and 2 depict the correlation between students' actual knowledge and their knowledge perception, which indicates a higher knowledge perception than the actual knowledge in both questionnaires. The results indicate that the correlation between the knowledge perception and actual knowledge is corrected by the end of the course (Final-Questionnaire), which is due to higher knowledge achieved by the end of the course. However, the knowledge perception remains at a high level.

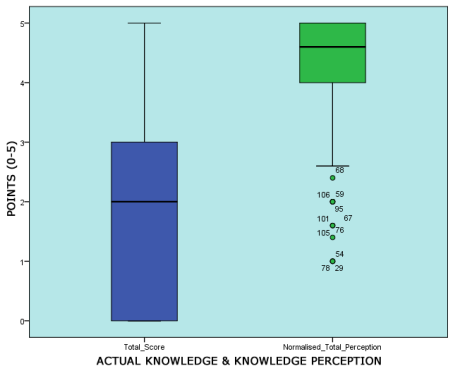


Figure 1: Correlation between students' actual knowledge and their knowledge perception. Intro-Questionnaire (course start).

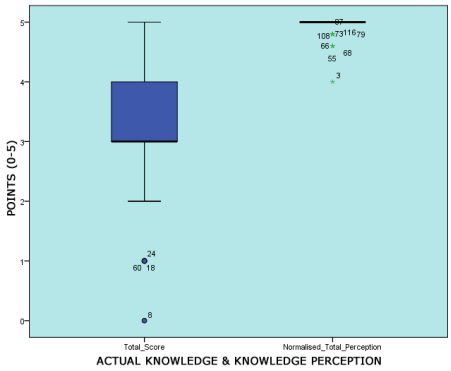


Figure 2: Correlation between students' actual knowledge and their knowledge perception. Final-Questionnaire (course end).

Table 2 reports on the ranks of the performed Wilcoxon signed-rank test. There were 100 out of 107 participants at the Intro-Questionnaire who assessed their knowledge higher than their actual knowledge was. On the contrary, only two participants reached the opposite results and only five assessed their knowledge correctly. The Final-Questionnaire results showed a slight increase in correctly assessed knowledge. There were 100 out of 116 participants at the Final-Questionnaire who assessed their knowledge as being higher than their actual knowledge was. On the contrary, only one participant reached the opposite result, while 15 assessed their knowledge correctly. A further indication of wrong knowledge perception can be deduced from the mean of the scored results. The mean of the perception score during the Intro-Questionnaire is 4.095, while the mean for the test score stood at 1.74. In addition, the means for the Final-Questionnaire were 4.957 and 3.35, respectively. We conclude that students overestimated their knowledge of CLR concepts throughout the entire course.

Table 2: Cases of knowledge perception scores versus actual knowledge scores.

Experimental instrument	Related Samples		N	Mean Rank	Sum of Ranks
Intro-Questionnaire	Percep. score - Test score	Negative Ranks	2 [a]	17.25	34.5
		Positive Ranks	100 [b]	52.19	5218.5
		Ties	5 [c]		
		Total	107		
Final-Questionnaire	Percep. score - Test score	Negative Ranks	1 [a]	1	1
		Positive Ranks	100 [b]	51.5	5150
		Ties	15 [c]		
		Total	116		

a Perception score <Test score
b Perception score >Test score
c Perception score = Test score

Conclusions regarding RQ2: Students overestimated their knowledge of CLR concepts throughout the entire course. The correlation between the students' knowledge perception and actual knowledge is corrected by the end of the course, due to higher knowledge reached by the end of the course. However, the knowledge perception remains at a high level.

3.2 Knowledge perception and notation

Additionally, we analyzed the results of the students' knowledge perception and actual knowledge considering the notation used in the learning process. Normalized results of students' self-assessment of their knowledge and results of our assessment of their knowledge was summarized and used to assess the students' perception of knowledge in terms of the dependence of the notation. The range of the summed score is thus 1 - 10. As the summed score approaches the extremes, the students were better able to assess their knowledge. It means that their perception of their knowledge and their actual knowledge were very close. On the contrary, the closer the results were to the middle, the more students incorrectly assessed their knowledge. It means that they either overestimated or underestimated it. For example students could assess their knowledge as high and reach five points for the perception and also score all five points on the test, thus collecting ten points. On the contrary, students could assess their knowledge as high, but reach a minimum or even none points on the test, thus scoring five points in total. The analysis of the impact of the notation was based on the data gathered from the Final-Questionnaire only, because the impact of the notation can only be seen after the notation was used in the learning process. Table 3 reports on the results of the Mann-Whitney U test for independent samples. We used the Mann-Whitney U test in order to compare differences between two independent groups (students using

Table 3: Correlation of summed perception and test score and influencing factor (notation used).

Exper. instr.	Independ. variable	Depend. variable	N	Asymp. Sig.	Decision
Final-Quest.	Notation	Summed perc. and test score	116	**0.008****	**Reject the null hypothesis**

*Significant at 5%; **Significant at 1%

Bachman or Barker notation) and the dependent variable (students' summarized test score and normalized perception score), while the groups are not normally distributed. The Shapiro-Wilk test of normality indicated that data significantly deviates from a normal distribution (p-value below 0.05).

The Mann-Whitney U test returns an asymptotic significance lower than 0.01 for the notation variable, therefore rejecting the related null hypothesis. The null hypothesis states that the distribution of the summed score is the same across categories of both Bachman and Barker notations. Considering the results, there is a statistically significant difference between the summed results scored by notation used in the learning process. There were 68 out of 116 students who used the Barker notation during the learning process, and their summed mean score stood at 8.1. The Bachman notation was used by 48 students, whereby their summed mean score was 8.608. According to Figure 3, it is evident that there were more students who used the Bachman notation and better assessed their knowledge.

Figure 3 depicts the correlation between the summed perception score and test score, and the notation used during the learning process.

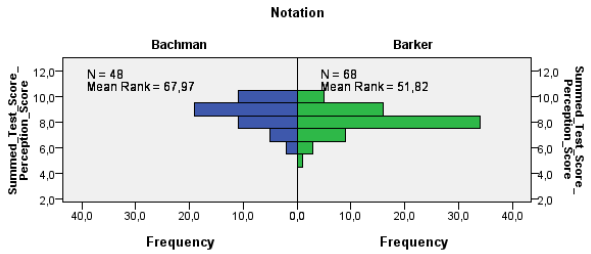


Figure 3: Summed perception score and test score in correlation with the notation used during the learning process.

On the contrary, there were students who used the Barker notation with a summed score of five which indicates the worst assessment of knowledge. We conclude that students who used the Bachman notation in the learning process better evaluated their knowledge than students who used the Barker notation.

Conclusions regarding RQ1: Bachman notation positively influences students' ability of knowledge self-assessment. By the course's end, the difference between knowledge perception and actual knowledge lowers.

4. CONCLUSIONS

The paper reported on the results of an experimental study aimed at analyzing the influence of notation used for the conceptual design on students' knowledge perception. The study continues on the work already presented in [7], while reporting on students' knowledge perception being higher than the actual knowledge.

We examined whether students' perception of knowledge is in accordance with their actual knowledge of CLR concepts. The results confirm that their perception is higher than the actual knowledge throughout the entire learning process. By the end, their knowledge increases and perception remains at a similar level as at the beginning. Additionally, the results prove that students who used the Bachman notation during the learning process were able to better estimate their knowledge. In the future we plan to analyze the correlation between students' educational background and their success rate while learning the CLR concepts on the higher education degree level.

5. ACKNOWLEDGMENTS

The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

6. REFERENCES

[1] A. Al-Shamailh. An Experimental Comparison of ER and UML Class Diagrams. *International Journal of Hybrid Information Technology*, 8(2):279–288, 2015.

[2] H. C. Chan, K. K. Wei, and K. L. Siau. Conceptual level versus logical level user-database interaction. In *ICIS 45. Proceedings*, pages 29–40, 1991.

[3] T. M. Connolly and C. E. Begg. A Constructivist-Based Approach to Teaching Database Analysis and Design. *Journal of Information Systems Education*, pages 43–53, 2005.

[4] R. Dargie and A. Steele. Teaching Database Concepts using Spatial Data Types. In *In Proceedings of the 4th annual conference of Computing and Information Technology Research and Education New Zealand*, pages 17–21, 2013.

[5] C. Domínguez and A. Jaime. Database design learning: A project-based approach organized through a course management system. *Computers & Education*, 55(3):1312–1320, 2010.

[6] D. C. Hay. A comparison of data modeling techniques. *Essential Strategies, Inc*, pages 1–52, 1999.

[7] A. Kamišalić, M. Heričko, T. Welzer, and M. Turkanović. Experimental Study on the Effectiveness of a Teaching Approach Using Barker or Bachman Notation for Conceptual Database Design. *Computer Science and Information Systems*, 15(2):421–448, 2018.

[8] H. C. Purchase, R. Welland, M. McGill, and L. Colpoys. Comprehension of diagram syntax: an empirical study of entity relationship notations. *International Journal of Human-Computer Studies*, 61(2):187–203, 2004.

[9] S. D. Urban and S. W. Dietrich. Integrating the Practical Use of a Database Product into a Theoretical Curriculum. *SIGCSE Bull.*, 29(1):121–125, 1997.

The Use of Standard Questionnaires for Evaluating the Usability of Gamification

Alen Rajšp
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
alen.rajsp@um.si

Katja Kous
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
katja.kous@um.si

Tina Beranič
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
tina.beranic@um.si

ABSTRACT
Usability has a significant impact on the satisfaction and frequency of use of a designed system. Nowadays, gamification and serious game approaches are implemented in software solutions to increase their usability. We present a literature review of 32 identified studies measuring usability, with established questionnaires in gamified systems and serious games. We identified 18 different questionnaires used for measuring usability, and found System Usability Scale to be the most widely used. An immense issue exists in the field, with only 22% of studies measuring usability actually describing or defining what usability is.

Categories and Subject Descriptors
H.5.2 [**User Interfaces**]: User-centered design

General Terms
Measurement, Experimentation, Standardization, Theory, Verification

Keywords
Usability Evaluation Method, Formal Questionnaires, Gamification, Serious Games

1. INTRODUCTION
In recent years, gamification has become an essential part of varieties of domains, from Education to Medicine. It is used for facilitating the use of developed products. They cannot achieve its purpose if the usability of the product is inadequate. Therefore usability evaluation should present a crucial step of development.

Solutions utilising (1) gamification or (2) serious game approach should be evaluated separately, due to them being inspired by games which have very specific (and different) natures. The primary function of games is to entertain through experience whereas serious games and gamification have some intended useful purpose [10]. Because gamification and serious games' approaches utilise elements from games, this leads to solutions where even other needs of solutions intended for an audience are being met to varying degrees. In solutions this causes an increase of user satisfaction.

In the web area of expertise, only 18% of the reviewed papers in [7] present usability evaluation methods relying on the standardised definitions of usability. Fernandez et al. [7] found that 59% of the reviewed papers reported end-user-based usability testing, while 35% of the reviewed papers used the inquiry methods (such as focus group, interviews, questionnaires and surveys). Based on these facts, this research focuses on inquiry methods, more specifically on technique questionnaires, and investigates which standard questionnaires are used most commonly for usability evaluation in The Gamification domain. Within the presented paper, we focus on the research question: *Which standard questionnaires are used for evaluating the usability of gamification?* Using a literature review, we study the use and popularity of established usability questionnaires in the Gamification domain.

A similar study, made by Yáñez-Gómez et al. [18], presents the review of academic methods for usability evaluation of serious games. The scope of the study is broader, aiming at finding the preferred approach for evaluating the usability of games. As the results show, standard questionnaires are the second most used technique applied in post-use analysis [18]. They mention three questionnaires in use, but detailed analysis is not provided. Also, in comparison to the presented study, our search string differs. Another review is presented by Calderón and Ruiz [5], also covering the domain of Serious Games' Evaluation. One of the research questions concerned evaluation techniques, and discovered that questionnaires are the most commonly used, but the categorization or detailed analysis of the used questionnaires was not provided.

The paper is structured as follows. We start by presenting the research background covering usability evaluation and gamification, we continue by presenting and discussing the results of the literature review. We close our paper by presenting the conclusions reached by our review.

2. USABILITY EVALUATION
The term usability represents a combination of several properties and attributes [13]. Regardless of the variety of definitions by different authors [1, 3, 9, 13, 15, 17], Jeng [12] states that Nielsen and ISO 9241-11 definitions are the most widely cited. ISO 9241-11 defines usability as "the extent to which a product can be used by a specified user to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [11], while Nielsen [15] defines

usability as an aggregation of five attributes: Learnability, efficiency, memorability, errors and satisfaction.

The usability evaluation method is defined as "a procedure, composed of a set of activities for collecting usage data related to end user interaction with a software product, and/or how the specific properties of this software product contribute to achieving a certain degree of usability" [7]. According to Battleson et al. [2], the usability evaluation methods are classified into three categories: (1) Inquiry methods (such as focus group, interviews, questionnaires and surveys), (2) Formal usability testing (such as interactions with a website by performing tasks) and (3) Inspection methods (such as heuristic evaluation, cognitive walk-through, pluralistic walk-through and formal inspection). The first two categories involve real-users, while inspection methods are based on reviewing the usability aspects of web artifacts, which have to comply with established guidelines, and are performed by expert evaluators or designers [7].

3. GAMIFICATION

Gamification is the use of design elements characteristic for games in non-game contexts [6]. Gamification should not be confused with serious games. Whereas the goal of introducing gamification is influencing learning related behaviuors and attitudes without providing knowledge, the use of serious games should influence learning and provide knowledge by the experience itself [14]. Another way to compare gamification and serious games is that gamification represents using only parts (game elements) from games, while serious games represent the whole immense gaming experience [6].

4. EVALUATING THE USABILITY OF GAMIFICATION

4.1 Research

Our research aims to find available standard questionnaires used for evaluating the usability of gamification. Using the following search string *"usability" AND ("gamification" OR "serious games" OR "educational games")* we conducted a search in the following digital libraries: *ScienceDirect*, *IEEE Xplore*, *ACM Digital Library* and *Sage journals*. Determined inclusion and exclusion criteria guided the study selection process. We considered the papers evaluating usability with the help of established and well-known questionnaires. Therefore, we excluded primary studies using ad-hoc questionnaires.

After the review process, we selected *33 primary studies*. The list of primary studies we used as input into the data extraction and data synthesis step is available at: *https://tinyurl.com/CSS2018-IJS*. 26 out of 33 primary studies are conference papers, whereas seven papers are journal articles. Figure 1 shows the number of primary studies by year of publishing. We selected 23 primary studies from the IEEE Xplore digital library, six from the ACM Digital Library, three from ScienceDirect and one from Sage journals.

4.2 Results

Within data extraction, we focused on two main areas. First, we searched for used definitions of usability, since the latter was evaluated in the analysed studies. Extracted data

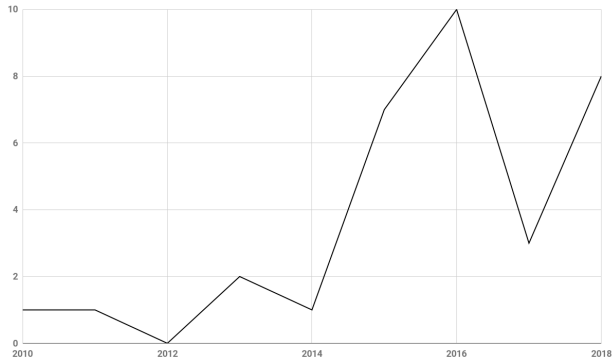


Figure 1: Primary studies by years

from selected primary studies showed that only seven primary studies (22%) defined and described the term of usability. Two of them indicated usability as a concept (S5, S10), while five researches treated usability as construct, namely two studies (S11, S21) used Nielsen's definition, one research (S4) used the ISO definition, one research (S18) described usability as "ease of use of the game", while study S25 defined usability similar to ISO, but expanded the definition with two new concepts ("simple" and "operating with ease"). The remaining studies (78%) used the term usability without providing the meaning of usability.

Studies are classified by domain in Table 1. Over half (56%) of all studies were from the field of Health and Medicine. Most of the studies from the domain addressed (1) Training of health care personnel (S8, S17, S18), (2) Rehabilitation and exercise for patients (S3, S6, S7, S16) and (3) Assessing patients (S1). The second most popular domain (37%) was Education and Learning. All other identified domains had only 1 study per domain.

Domain	**Primary studies**
Agriculture	S27
Business Intelligence	S5
Computer Science	S5
Education & Learning	S2, S8, S10, S13, S14, S16, S17, S18, S23, S28, S29, S31
Entertainment	S4
Health & Medicine	S1, S3, S6, S7, S8, S11, S12, S16, S17, S18, S19, S20, S21, S22, S24, S29, S30, S32
Social Science	S25
Task Management	S9
Travel	S15

Table 1: Domain

We continued the data extraction by identifying standard questionnaires used for usability evaluation. We followed the explanation provided by Yáñez-Gómez et al. [18], which states that *standard questionnaires are the ones that are validated statistically.* Table 2 presents used questionnaires in connection with primary studies. The majority of studies evaluated usability by using the System Usability Scale

(SUS). It was used in 78% of primary studies. Although Technology Acceptance Model (TAM) is used in the model-driven analysis for measurement of users' acceptance and usage of technology and it is not classified as a standard questionnaire for usability evaluation, it was used for assessment of gamification in four primary studies. On the other hand, Game Experience Questionnaire (GEQ), Task Load index (TLX), Game Engagement Questionnaire (GEQ), Post-Study System Usability Questionnaire (PSSUQ) and Net Promoter Score (NPS) are each used in two primary studies. We extracted other questionnaires that are used only in one primary study, such as Presence Questionnaire (PQ) and Software Usability Measurement Inventory (SUMI). To achieve

Questionnaire	Primary studies
System Usability Scale (SUS)	S1, S3, S6, S7, S10, S11, S12, S14, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28, S29, S30, S31, S32
Technology Acceptance Model (TAM)	S2, S3, S11, S20
Game Experience Questionnaire (GEQ)	S1, S4, S30
Task Load index (TLX)	S1, S22
Game Engagement Questionnaire (GEQ)	S11, S18
Post-Study System Usability Questionnaire (PSSUQ)	S8, S15
Net Promoter Score (NPS)	S31-S32
User Engagement Scale (UES)	S5
Computer System Usability Questionnaire (CSUQ)	S13
Software Usability Measurement Inventory (SUMI)	S7
Intrinsic Motivation Inventory (IMI)	S16
User Interaction Satisfaction (QUIS)	S18
Presence Questionnaire (PQ)	S10
Usefulness, Satisfaction, and Ease of use (USE) Questionnaire	S9
Pick-A-Mood (PAM)	S10
Technology Affinity - Electronic Devices (TA-ED) Questionnaire	S20
Game User Experience and Satisfaction Scale (GUESS)	S19
Differential Emotions Scale (DES)	S10

Table 2: Standard questionnaires in use

a comprehensive usability evaluation, it is crucial that measurement instruments used are utilised appropriately according to the attribute they are measuring. 41% (13/32) of primary studies (S6, S12, S17-S19, S25-S30, S32) used the most established questionnaire SUS for measuring usability, but did not define the measured attribute in their research. Table 3 presents the connection between the used questionnaires and measured attributes that were measured at least in two primary studies. The most frequently measured attributes were "ease of use" and "usability" and both were used in six primary studies. In all cases, the attribute "usability", was measured with SUS, while the attribute "ease of use" was measured with three different questionnaires: SUMI (S7), USE (S9) and TAM (S2, S3, S11, S20). The second most frequent measured attribute was attribute "usefulness". In three primary studies (S2, S11, S20), it was treated and determined as one of the two factors defined in TAM, while, in one case, it was measured with USE (S9) and PSSUQ (S15). The attribute "satisfaction" was the third most commonly used attribute measured by two different questionnaires: SUS (S21, S22, S31) and USE (S9).

Measured attribute	Questionnaires
Ease of use	SUMI (S7), USE (S9), TAM (S2, S3, S11, S20)
Usability	SUS (S10, S11, S16, S23, S24, S31)
Usefulness	TAM (S2,S11,S20), USE (S9), PSSUQ (S15)
Satisfaction	USE (S9), SUS (S21, S22, S31)
Flow	GEQ (S1, S4, S11)
Learnability	SUMI (S7), USE (S9)
Competence	GEQ (S1, S4)
Overall	CSUQ (S13), SUMI (S7)
Quality of Information	CSUQ (S13), PSSUQ (S15)
Quality of interface	CSUQ (S13), PSSUQ (S15)

Table 3: Connection between the measured attributes and used questionnaires

The most popular devices on which developed/proposed solutions were run were computers (62%), virtual reality equipment (22%) and mobile devices (16%) as seen in Table 4.

Device	Primary studies
Computer	S1, S2, S4, S5, S6, S7, S8, S10, S11, S13, S14, S16, S21, S22, S23, S25, S26, S28, S29, S31
Customised system	S19
Mobile device	S9, S12, S15, S20, S27
Smart TV	S3
Virtual reality	S10, S15, S17, S18, S24, S30, S32

Table 4: Devices on which the studied system runs

4.3 Discussion

An extensive collection of standard questionnaires were found for evaluating the usability of gamification, with System Usability Scale (SUS) as the prevailing choice (84% of all studies). Since SUS is a well-known questionnaire, which is easy to perform and analyse, this is not a surprise. As SUS was developed for providing a subjective assessment of usability [4], its extensive use is even more understandable. The majority of researchers that used SUS in their studies did not

quote explicitly which attribute of usability was measured; the remaining studies, where the SUS were used, defined two different attributes that can be measured with SUS. The first attribute was "usability" and it is in accordance with description of SUS usage purpose [4], while the second one was "satisfaction", which is recommended by the ISO/TS 20282-2:2013 [8] Standard, where the SUS is defined as a questionnaire for measuring satisfaction.

Another aspect is also if standard usability questionnaires can evaluate the usability of gamification adequately. Shegawa et al. [16] claims that the SUS questionnaire is a verified instrument for measuring usability in the Serious Games domain. Technology Acceptance Model (TAM) is widely used in the Information System domain to investigate how accepted the use of technology is among their target users. Although it is not classified as a standard questionnaire for usability evaluation, but rather as a model combining constructs ease of use and usefulness, it was the second most used measuring instrument for usability evaluation in reviewed literature. On the other hand, it is also seen that questionnaires, like Game Experience Questionnaire (GEQ) and Game Engagement Questionnaire (GEQ), that originate from Gaming domain, are nowadays used to evaluate the usability of gamification. Therefore, the fusion of two fields is perceived.

5. CONCLUSION

The paper presents conducted literature review which was aimed at finding standard questionnaires used for usability evaluation of gamification and serious games. We found that the majority (84%) of studies evaluate usability using a System Usability Scale (SUS), though some other questionnaires were also detected and used independently, or in combination with SUS. We, as prospective researchers, can determine only in a minority of cases what primary studies were measuring, because only 22% of primary studies measuring usability defined or described what usability is. That is an immense issue on validity of their measurements of usability, since multiple definitions of it exist. We propose that methods for measuring usability in the field of Gamification and Serious Games should be formalised in the future. Although researchers are already using standardised methods for measuring usability, research should also present what *usability* means for them, what they are *measuring*.

6. ACKNOWLEDGMENTS

The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

7. REFERENCES

[1] A. Abran, A. Khelifi, W. Suryn, and A. Seffah. Usability meanings and interpretations in iso standards. *Information and Software Technology.*, 11(4):325–338, Aug 2003.

[2] B. Battleson, A. Booth, and J. Weintrop. Usability testing of an academic library web site: A case study use of academic library web. *J. Acad. Librariansh.*, 27(3):325–338, 2001.

[3] T. Brinck, D. Gergle, and S. D. Wood. *Designing Web Sites that Work: Usability for the Web.* Morgan Kaufmann, San Francisco, 2002.

[4] J. Brooke. Sus: A quick and dirty usability scale, 1996.

[5] A. Calderón and M. Ruiz. A systematic literature review on serious games evaluation: An application to software project management. *Computers & Education*, 87:396–422, 2015.

[6] S. Deterding, D. Dixon, R. Khaled, and L. Nacke. From game design elements to gamefulness: Defining gamification. *Proceedings of the 15th International Academic MindTrek Conference on Envisioning Future Media Environments - MindTrek '11*, pages 9–11, 2011.

[7] A. Fernandez, E. Insfran, and S. Abrahão. Usability evaluation methods for the web: A systematic mapping study. *Inf. Softw. Technol.*, 53(8):789–817, Aug 2011.

[8] I. O. for Standardization. ISO/TS 20282-2:2013 Usability of consumer products and products for public use - Part 2: Summative test method, 2013.

[9] E. Furtado, J. J. V. Furtado, F. Lincoln Mattos, and J. Vanderdonckt. Improving usability of an online learning system by means of multimedia, collaboration, and adaptation resources. In *Usability Eval. Online Learn. Programs*, pages 69–86, October 2003.

[10] C. Girard, J. Ecalle, and A. Magnan. Serious games as new educational tools: how effective are they? A meta-analysis of recent studies. *Journal of Computer Assisted Learning*, 29(3):207–219, 2013.

[11] ISO. *Standard 9241: Ergonomic Requirements for Office Work with Visual Display Terminals (VDT)s, Part 11. Guidance on Usability.* 1998.

[12] J. Jeng. What is usability in the context of the digiral library and how can it be measured? *Information Technology and Libraries*, 24(2):47–56, Nov 2005.

[13] Z. Kılıç Delice, Elif Güngör. The usability analysis with heuristic evaluation and analytic hierarchy process. *Int. J. Ind. Ergon.*, 39(6):934–939, Nov 2009.

[14] R. N. Landers. Developing a Theory of Gamified Learning: Linking Serious Games and Gamification of Learning. *Simulation & Gaming*, 45(6):752–768, 2014.

[15] J. Nielsen. *Usability Engineering.* Academic Press, San Diego, 1993.

[16] R. Shewaga, A. Uribe-Quevedo, B. Kapralos, K. Lee, and F. Alam. A Serious Game for Anesthesia-Based Crisis Resource Management Training. *Entertainment Computing*, 16(2):6:1–6:16, apr 2018.

[17] G. Tsakonas and C. Papatheodorou. Exploring usefulness and usability in the evaluation of open access digital libraries. *Information Processing & Management*, 44(3):1234–1250, May 2008.

[18] R. Yáñez-Gómez, D. Cascado-Caballero, J.-L. J.-L. Sevillano, R. Yanez-Gomez, D. Cascado-Caballero, and J.-L. J.-L. Sevillano. Academic methods for usability evaluation of serious games: a systematic review. *Multimedia Tools and Applications*, 76(4):5755–5784, Feb 2017.

Analyzing Short Text Jokes from Online sources with Machine Learning Approaches

Samo Šimenko
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
samo.simenko@student.um.si

Vili Podgorelec
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
vili.podgorelec@um.si

Sašo Karakatič
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
saso.karakatic@um.si

ABSTRACT
This paper presents the whole data mining process of analyzing jokes in Slovenian language gathered from various online sources. The gathering was done with the help of web scrapping system and the analysis was carried out on the gathered jokes to determine the properties of various types of jokes. In addition, with the help of various text-mining methods, we analyzed different types of jokes and built a machine learning model for classifying jokes into categories. These results are supplemented with the visualization of different categories and the interpretation of constructed machine learning classification models.

Categories and Subject Descriptors
H.4 [**Information Systems Applications**]: Miscellaneous;

I.2.m [**Artificial Intelligence**]: Miscellaneous;

General Terms
Machine Learning, Data Mining, MDS, SVC

Keywords
Data mining, Machine Learning, Joke analysis, Short text analysis, Text mining

1. INTRODUCTION
Due to the ever-advancing technology, opportunities are opening for analyzing all types of data, so we can make the most of this and use it for our benefit. By studying and examining various types of texts, scientists were already involved in the initial phases of textual analysis [8, 9, 10], but studying the meaning and connection of texts presents a rather new direction of research, where there is still a lot of room for improvement. While there has been a lot of work done on various short text types, i.e. tweets [12], reviews [11], recipes [13] and others, there is a lack of research published on the topic of jokes analysis.

In our paper, we present a process of gathering, parsing and pre-processing jokes and applying various data- and text-mining techniques to extract patterns and new knowledge from jokes data. By semantic text processing, we identify more than just a sequence of symbols, we can assign them meaning, which can influence the classification of jokes. In our case, we undertook the processing of various jokes that we analyzed in order to determine how the categories of such texts are interconnected by their content and find out which categories of jokes share the most similar content. Based on the texts, we created a classification model for the classification of jokes into predefined categories.

The rest of the paper is structured in the following way. The following section presents the method for gathering and parsing jokes from the online sources. Third section presents the individual steps of data- and text- mining in details. It consists of machine learning method description, applications and techniques used in the process and the results itself. We finish up with the conclusion and the discussion on the topic of joke analysis with various data mining methods.

2. GATHERING AND PARSING OF THE JOKES FROM THE ONLINE SOURCES
In order to fulfill the set goals of analyzing jokes, we obtained these from various sources. Three different sources were used:

– From the first source, a web site called **VERZIVICI** [2], joker already classified into categories;

– Jokes from the second source **NAJVICI** [3];

– Jokes from a third source **MLADINSKI** [4].

For the data acquisition we developed a program in the Visual Studio IDE, using the C# programming langauge, which acquired jokes from the selected sources and saved them in a suitable text format. Due to the unstructured data of selected web resources, we used HAP (HTMLagilityPack) for processing. HAP is a HTML parser written in C# for reading/writing the DOM (Document Object Model) and supports plain XPATH or XSLT [1]. Using the HAP library and XPATH, we could easily access individual sections, which contained content known as a "joke".

Jokes from VERZIVICI, which were categorized when gathered, were manually entered, since the program for collecting jokes from different categories used the category name in the creation of a URL, which is used for scrolling between categories. For NAJVICI, we manually created a URL for gathering jokes so we can easily access all jokes on the site.

On the website MLADINSKI, jokes were already grouped and the jokes were sequentially recorded on one side of the web page. For the purpose of processing and subsequent manipulation, a simple VIC class was created, which contains two textual attributes of Text and Category. Both attributes can store values in string format, Text attribute is for raw text of a joke, Category is for type of category in which joke is categorized. When we were capturing blank spaces, we encountered redundant badges before text and between texts. Also, unreadable machine records were created instead of symbols due to coding. All badges with associated symbols and non-nominal groups of words, which were created instead of symbols, were manually entered into the program and then programmatically removed.

As a result of obtaining and processing the data from the selected sources, we received the data, which are used as the basis below:

– VERZIVICI [2] – 13 categories, a total of 1729 jokes,

– NAJVICI [3] – a total of 297 jokes, and

– MLADINSKI [4] – a total of 145 jokes.

We have saved the acquired data in the CSV format. Due to the characteristics of the CSV format, the comma symbol "," was changed to the XX symbol, addressed below, because comma in CSV represents a separator between lines, in jokes commas can have different meaning. All of the jokes were in Slovenian language, so this had to be taken into consideration during the text analysis.

3. DATA ANALYSIS

In this section, we will present the methods and techniques for analyzing the jokes and the results of these analysis. The whole process of cleaning, preprocessing, and the analysis itself was done with the Python programming language, and its libraries.

3.1 Cleaning and preprocessing the data

As mentioned, we use the Python programming language to process data in which you can simply import information in a CSV format using the Pandas library [5]. Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language [5, 14]. The imported data is then appropriately structured using the DataFrame class with the following columns (attributes):

– Index,

– Category, and

– RawText.

The XX symbols are also removed and replaced with the comma symbol ",". From the text, we also removed stop-words, which is a list of common words that do not carry any semantic meaning and information. Stop words occurred in texts in high frequency but are of little significance and consequently uninteresting. A sample of stop words in Slovenian language are the following:

“in” (En. and),	“ali” (En. or),
”je”(En. is),	”za” (En. for),
”to” (En. this),	”na” (En. on),
”to” (En. this),	”ti” (En. you),
”ko” (En. when),	”bi” (En. would),
”ne” (En. no),	”da” (En. yes),
”že” (En. already),	”le” (En. only).

In addition, the punctuations were removed, so the resulting text was in the form of one sentence without most common stop words.

From the resulting text, we built a representation of every joke in the format appropriate for the analysis. We used the method of counting the frequency of individual words called word frequency. This number was normalized by the word frequency of the word in all categories, so the more common words got the lower score and the less common and maybe unique words got higher score. This process is called tf-idf (term frequency-inverse document frequency) and is a common word scoring method in text mining [17]. The new dataset was built in such way, that all of the identified words represented one attribute of the joke, and the corresponding value of that attribute is the tf-idf score of that word in that joke.

3.2 Classification of jokes in the predetermined categories

We used the classification machine learning technique in order to construct a model of classification that would learn how to classify yet unseen jokes to one of the predetermined categories. This can be useful if one would want to automate joke categorization on an online joke portal without any need for human intervention. The classification is a supervised machine learning method, which means that machine learns to classify jokes from the already solved (classified) examples [15].

There are numerous different classification algorithms [18], but for our case we used the Support Vector Machine (SVM) classifier, developed by Vapnik in 2000 [16]. This method learns the boundaries that separate instances (jokes in our case) from one category to another, by finding a linear separation border called hyper-plane that has a maximum distance from the entire instance set, which is called the maximum margin. The instances that are closest on the hyper-plane (on the hyper-plane itself) are called support vectors. This SVM method also uses a kernel trick [19], which maps the attribute space of the classification instance to a higher dimensional space. In our case, we used a linear kernel, which uses a liner function to transform the attributes in such a way, that the margin of the hyper-plane is maximized.

We used the implementation of SVM from the library liblinear [20], which has high flexibility in the choice of penalties and loss functions and should scale to large numbers of samples. This library supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme [6].

Upon preliminary data preparation, the whole joke dataset is divided into train and test sets, where the training set is used to build the SVM classification model, and the test set is used to test the quality of the model – the ability to correctly classify yet unseen jokes. In our experiment, we applied stratified sampling to split the data and used 60% of data for training test and the rest 40% for the test set. The results of the experiment show, that the resulting classification model classifies test jokes with 61% accuracy. The classifier has correctly classified more than half of jokes into their proper category out of 13 possible categories.

The default classification of instances in one of 13 categories would result in only 0.08 accuracy, so our resulting classifier improves the default classifier significantly. This represents a high percentage of precision as was not foreseen at first glance. Additionally, we also manually examined some of the jokes that were misclassified. Interestingly, although the predicted categories were not correct, several of the examined jokes would fit well into the predicted category as well, as the semantics of a joke is not always monolithic.

3.3 Word frequency analysis and visualization

From the dataset of jokes, with attributes of individual word’s tf-idf scores, we built word cloud diagrams for every category of the joke. The word clouds were made with the help of libraries matplotlib [21] and wordcloud for the Python programming language. In the word cloud, the most common words (or rather those with higher tf-idf scores) are written in larger font, while those with lower frequency (lower tf-idf scores) are written in smaller font. The color of the words only serves to make words more differentiable and thus improves the readability of word clouds.

Also, these word cloud show which highly informative words (non-stop words) are common for each category and can be used for manual classification, this way we can check whether a joke, which reads: “pride nekega dne k janezkovemu očetu domov nek njegov nadležen prijatelj tone potrka vpraša dober dan oče doma janezek tone ja kje janezek vem grem vprašat” was appropriately classified into a category (the original category is called “janezek”, “Solski” was predicted). As we can see in Figure 1, our model correctly decided to classify the joke in the category “Solski”, because the

word “janezek” prevails in this category and is the dominant word in the content of the joke.

Figure 1: Word-clouds for ten joke categories.

3.4 Hierarchy of the categories

With the help of the scipy [22] Python library, we also built a dendrogram of relations between the categories using a hierarchical clustering method, which is shown in the Figure 2. Here we also included the category from sources NAJVICI and MLADINSKI, so that we can visually display the content linkage between different categories. The dendrogram is a hierarchical diagram, which shows which terms (in our case joke categories) are closer together by putting the more similar categories closer together on the Y-axis. The more similar are the categories, shorter are the lines connecting these categories, and vice versa.

From the dendrogram we can see that the categories MLADINSKI (En. young ones) and SOLSKI (En. School ones) are most similar, since the school is usually visited by young people. Based on the names of the categories NAJVICI and Mesane sale (En. Random jokes), it can also be assumed that these categories are very similar. From the dendrogram we can also see that groups of categories marked by red and green connections are very different. We can conclude that this division can be attributed primarily to slang expressions, which are more commonly used in foreign jokes as well as older jokes.

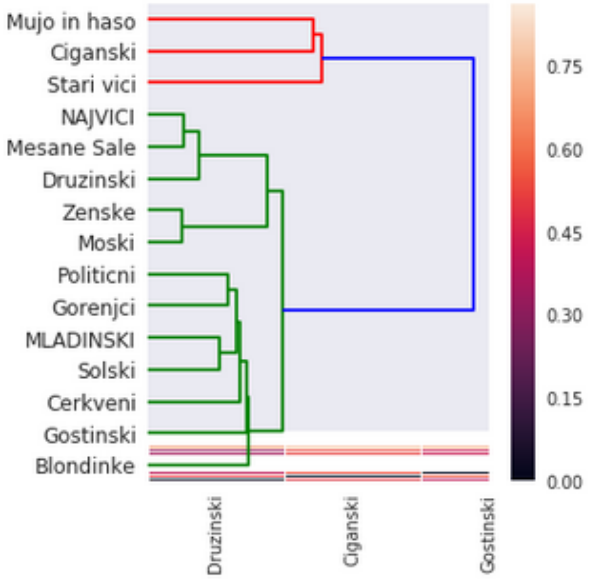


Figure 1: Hierarchical clustering of joke categories.

3.5 Multidimensional scaling

Multidimensional scaling (MDS) enables the visualization of the level of similarity of individual cases of a dataset by lowering the number of different attributes to only two. It refers to a set of related ordination techniques used in information visualization, in particular to display the information contained in a distance matrix [7]. By using the MDS in the sklearn.manifold[23] library and the mpl_toolkits.mplot3d[24] library, we can observe relations between categories even more efficiently, as shown in a 2D graph in the Figure 3. This plot shows which categories are closer together and which categories differ the most. Contrary to the dendrogram, we can see that “Mujo in Haso” are not so close to “Ciganski” and “Stari vici”, but these three categories differ the most from the rest.

Figure 2: 2D Multidimensional scaling plot, which shows the similarity of different joke categories

This shows the seclusion of three categories (a group of categories marked in a dendrogram with red color, which includes Stari Vici, Mujo in Haso and Ciganski) in relation to other categories. These make up a kind of circle around the categories "NAJVICI" and "Mesana Sale". Categories "NAJVICI" and "Mesana Sale" are the closest neighbors, which also suggests an exceptional similarity between the categories.With the help of Figure 3, we can see the relationship between categories even better; in the case of the categories "Moski" and "Zenske", we can see that according to their content, these two are very similar categories.

As depicted in the Figure 4 is a 3D graph of relations for use in further discussions for the show. By using the 3D graph (Graph 4), we can even more accurately determine the differences between the categories of texts. This display mode can turn out to be even more useful in a larger number of data and when looking for interesting patterns in these texts.

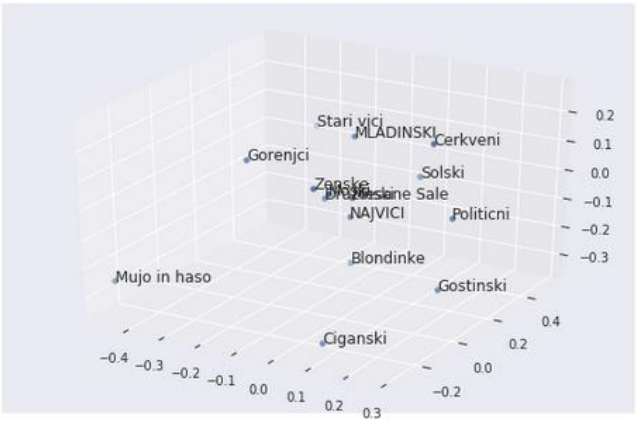


Figure 3: 3D Multidimensional scaling plot, which shows the similarity of different joke categories

4. CONCLUSION

This paper presents a use case of machine learning methods in the analysis of short texts in a form of jokes. We presented the process of gathering, cleaning and preprocessing the jokes, which was followed by the description of the analysis done with machine learning methods and various visualization techniques. We demonstrated how jokes could be automatically categorized in the predefined categories using the Support Vector Machine classification method. With two different visualizations: the dendrogram and the multidimensional scaling plot, we showed how different joke categories are similar one to another. With these methods, we demonstrated, how we could perform different comparisons, which can serve us in the further processing of data, and the connection of data between us is visualized in a useful and interesting way.

In this paper, we only analyzed the jokes in Slovenian language. For future work, we could compare jokes in different languages to find similarities and differences of jokes and their popularity across different languages and cultures.

ACKNOWLEDGMENTS

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0057).

REFERENCES

[1] http://html-agility-pack.net, Last visited: 20.8.2018

[2] http://www.verzi-vici.com, Last visited: 5.8.2018

[3] http://www.naj-vici.com, Last visited: 5.8.2018

[4] http://www.mladinska.com/, Last visited: 5.8.2018

[5] https://pandas.pydata.org, Last visited: 13.8.2018

[6] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html, Last visited: 13.8.2018

[7] https://en.wikipedia.org/wiki/Multidimensional_scaling,

Last visited: 20.8.2018

[8] Song, G., Ye, Y., Du, X., Huang, X. and Bie, S., 2014. Short text classification: A survey. Journal of Multimedia, 9(5), p.635.

[9] Chen, M., Jin, X. and Shen, D., 2011, July. Short text classification improved by learning multi-granularity topics. In IJCAI (pp. 1776-1781).

[10] Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H. and Demirbas, M., 2010, July. Short text classification in twitter to improve information filtering. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval (pp. 841-842). ACM.

[11] Dave, K., Lawrence, S. and Pennock, D.M., 2003, May. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In Proceedings of the 12th international conference on World Wide Web (pp. 519-528). ACM.

[12] Wang, Y., Liu, J., Qu, J., Huang, Y., Chen, J. and Feng, X., 2014, December. Hashtag graph based topic model for tweet mining. In Data Mining (ICDM), 2014 IEEE International Conference on (pp. 1025-1030). IEEE.

[13] Badra, F., Bendaoud, R., Bentebibel, R., Champin, P.A., Cojan, J., Cordier, A., Després, S., Jean-Daubias, S., Lieber, J., Meilender, T. and Mille, A., 2008, September. Taaable: Text mining, ontology engineering, and hierarchical classification for textual case-based cooking. In 9th European Conference on Case-Based Reasoning-ECCBR 2008, Workshop Proceedings (pp. 219-228).

[14] McKinney, W., 2012. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. " O'Reilly Media, Inc.".

[15] Friedman, J., Hastie, T. and Tibshirani, R., 2001. The elements of statistical learning (Vol. 1, No. 10). New York, NY, USA:: Springer series in statistics.

[16] Vapnik, V. and Mukherjee, S., 2000. Support vector method for multivariate density estimation. In Advances in neural information processing systems (pp. 659-665).

[17] Aizawa, A., 2003. An information-theoretic perspective of tf–idf measures. Information Processing & Management, 39(1), pp.45-65.

[18]http://en.wikipedia.org/wiki/Category:Classification_algorithms, Last visited 13.9.2018

[19] https://en.wikipedia.org/wiki/Support_vector_machine Last visited 13.9.2018

[20] https://www.csie.ntu.edu.tw/~cjlin/liblinear/, Last visited 13.9.2018

[21] https://matplotlib.org/, Last visited 13.9.2018

[22] https://www.scipy.org/, Last visited 13.9.2018

[23] http://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html, Last visited 13.9.2018

[24] https://matplotlib.org/2.0.2/mpl_toolkits/mplot3d/api.html

A Data Science Approach to the Analysis of Food Recipes

Tjaša Heričko
Faculty of Electrical Engineering and Computer Science
University of Maribor, FERI
Maribor, Slovenia
tjasa.hericko@student.um.si

Sašo Karakatič
Faculty of Electrical Engineering and Computer Science,
University of Maribor
Maribor, Slovenia
saso.karakatic@um.si

Vili Podgorelec
Faculty of Electrical Engineering and Computer Science,
University of Maribor
Maribor, Slovenia
vili.podgorelec@um.si

ABSTRACT
In this paper, we explore the correlation between cuisine and text-based information in recipes. The experiments are conducted on a real dataset consisting of 9,080 recipes with data science approaches focusing on enhancing cuisine prediction and providing a detailed insight on the characterization of food cultures. The findings suggest that information about ingredients is the most relevant predictor of cuisines, however, despite being less efficient, recipe name, preparation instructions, preparation time, skill level and nutritional facts can be considered as well.

Categories and Subject Descriptors
I.2.m [**Artificial Intelligence**]: Miscellaneous.
I.5.m [**Pattern Recognition**]: Miscellaneous.

General Terms
Algorithms, Measurement, Experimentation.

Keywords
Data Science, Machine Learning, Text mining, Classification, Food Recipes, Cuisines.

1. INTRODUCTION
In response to technological advancements and social changes in the last decades, the tendency to collect and store recipes only in cookbooks has changed. Numerous online recipe portals started to rapidly accumulate food-related content, with more and more recipes being published online daily. The growth in the amount of user-generated recipe data available on the Internet has raised several issues that researchers have been trying to address in recent years. The objective of this paper is to explore the correlation between cuisine and text-based information in recipes, including recipe name, list of ingredients, preparation instructions, preparation time, skill level, calories and nutritional information. The results of this study address the issue of automatic recipe cuisine categorization, making it easier to submit a new recipe and preventing possible additional noise in recipe database – this can be helpful for both the contributors as well as for the culinary website curators.

We conducted a series of experiments on a real dataset retrieved from BBC Good Food[1] consisting of 9,080 recipes from various cuisines with data science approaches focusing on the following: (1) Providing a detailed insight on the characterization of various food cultures. (2) Identifying necessary text-based information from recipes needed to perform well at cuisine prediction. (3) Enhancing cuisine prediction.

This paper is organized as follows. Section 2 gives a brief overview of related work. Section 3 presents the dataset used in our research. Section 4 describes the applied methodologies. Section 5 provides results of our research. Section 6 concludes the paper by summarizing the main results of our work.

2. RELATED WORK
The correlation between recipes and their cuisines has been the subject of several recipe analysis related research. Mostly, there have been previous studies conducted on classifying recipes into respective cuisines based on ingredients. H. Su et. al. [1] evaluated data collected from Food[2] and used the techniques of associative classification and support vector machine to classify 226,025 recipes to one of six cuisines, using ingredients as inputs, with a precision and recall of about 75 %. The researchers in [2–8] further studied cuisine-ingredient connection, using 39,774 recipes from twenty cuisines provided by Yummly[3]. Similar studies were conducted on data from Epicurious [9], Epicurious and Menupan[4] [10] and Food, Epicurious[5] and Yummly [11]. A variety of machine learning algorithms, including k-means [2, 9], random forest classifier [2, 5, 6, 8, 9, 10], support vector machine [3, 5, 6, 7, 10, 11], logistic regression [4, 5, 6, 10, 11] and naive Bayes [5, 6, 7, 9, 10, 11], were used in these studies. From several tested algorithms, linear support vector machine, reaching up to 80,9 % accuracy in [7], was found to be the most efficient for this cuisine prediction task based on ingredients.

Other studies focused on the importance of other information extracted from recipes for cuisine prediction. H. Kicherer et. al. [12] evaluated the use of ingredients and preparation instructions for cuisine prediction, conducted on recipes from German website Chefkoch[6]. The study revealed that ingredients alone are as good an indicator as the recipe instructions. Whereas a combination of information from both – nouns from the instructions and the list of ingredients – performs better. T. Ozaki et. al. [13] also demonstrated that, based on Japanese recipes from Cookpad Data[7], certain sets of ingredients and preparation actions deeply correspond to cuisine types.

Previous studies have already noted that ingredients reveal important information about cuisines and that predicting cuisines

[1] https://www.bbcgoodfood.com/

[2] https://www.food.com/
[3] https://www.yummly.com/
[4] https://www.menupan.com/
[5] https://www.epicurious.com/
[6] https://www.chefkoch.de/
[7] https://cookpad.com/

based on the ingredients is possible. Though, to our knowledge, few researchers have considered using additional text-based information from recipes, for instance, preparation instructions, preparation time and nutrition facts, as possible attributes in cuisine prediction. Therefore, there is little understanding of how they are related to cuisine types. In contrast to the work presented above, we performed a richer analysis of recipes with a wider range of attributes extracted from recipes, whereas the dominant approach appears to deal only with ingredients as attributes.

3. DATASET

Our research was conducted on the crawled data collected from an online food recipe portal BBC Good Food. A dataset of 9,429 recipes was scraped with Python[8], using Scrapy framework[9] and CSS selectors, in June 2018.

For each recipe, the following information was provided: recipe name, cuisine, list of ingredients, preparation instructions, preparation time, skill level and nutrition facts, including the amount of calories, total fat, saturated fat, total carbohydrate, sugars, protein, fiber and salt per serving. More details are presented in Table 1.

Table 1. Characteristics of text-based information in recipe

Information	Data Type	Description
Recipe name	Unstructured	Arbitrary string described in natural language.
Cuisine	Categorical	One of 45 cuisine types.
List of ingredients	Unstructured	Arbitrary string depicting needed ingredients for preparation, each ingredient normally consisting of an ingredient type, an amount and a unit.
Preparation instructions	Unstructured	Step-by-step instructions for preparation using ingredients described in natural language.
Preparation time	Numerical	A number representing time measured in minutes needed for preparation.
Skill level	Categorical	One of 3 difficulties: easy, more effort or a challenge.
Nutrition facts	Numerical	A number representing nutrition per serving measured in kcal for calories intake or in grams for fat, saturated fat, carbohydrate, sugars, protein, fiber and salt.

4. METHODOLOGY

The methodology in this paper was implemented in Jupyter notebook environment[10] running Python code and using a combination of Python libraries comprising pandas[11], scikit-learn[12], NLTK[13], seaborn[14], matplotlib[15] and wordcloud[16].

[8] https://www.python.org/
[9] https://scrapy.org/
[10] http://jupyter.org/
[11] https://pandas.pydata.org/
[12] http://scikit-learn.org/
[13] https://www.nltk.org/
[14] https://seaborn.pydata.org/
[15] https://matplotlib.org/
[16] http://amueller.github.io/word_cloud/

4.1 Data Preprocessing

For the dataset to be feasible for the analysis, preprocessing was performed on the raw scraped data.

During the data cleaning step, missing values and duplicates were resolved by removing these recipes from the original dataset, leaving a subset of 9,080 recipes.

The original dataset included 45 cuisine categories, many of them only consisted of few recipes. In the next step of data preparation, based on the findings of previous researches of cuisines being location-dependent [14], we combined smaller cuisines into bigger regional cuisine categories (e.g. Balinese, Thai, Vietnamese and Indonesian into Southeast Asian cuisine) and therefore reduced cuisine categories to the following 13: African, Middle Eastern, South Asian, Southeast Asian, East Asian, Oceanic, American, Latin American, Western European, Northern European, Central European, Eastern European, Mediterranean.

As highlighted in Table 1, preparation time and nutrition facts are numerical, cuisine and skill level are categorical, whereas recipe name, list of ingredients and preparation instructions are described in natural language. For all of them, additional preprocessing was needed prior to conducting analyses. Numerical attributes were standardized, considering certain algorithms used in our research are sensitive to varied number scales and intervals used [15]. As scikit-learn algorithms only work on numerical data, categorical data needed to be encoded as numerical. This was done by converting categorical data into dummy variables [16]. For unstructured data to be used for classification, several more text preprocessing methods were needed: tokenization, stop word removal, stemming and tf–idf term weighting. Tokenization is the process of segmenting a text into identifiable basic linguistic units called tokens, such as words and punctuation [17]. For better processing, all tokens were converted to lowercase. Stop words are frequently used common words, such as ‘and’, ‘the’ and ‘this’. Because their presence in a text fails to distinguish it from other texts and are therefore not useful in classifications, they were removed before further processing [18]. We also made a custom list of stop words, where we included numbers that represent amounts and words that represent units, e.g. ‘2’ and ‘tbs’, that would not be of value in the analysis. The same applies to punctuation, therefore they were filtered out as well. Next, stemming using the Porter stemming algorithm, the process of removing morphological affixes from words, which conflate variant forms of a word into a unified representation [19], was performed. Lastly, for words counts being suitable for usage by a classifier, tf–idf transform was conducted. Tf–idf, short for term-frequency times inverse document-frequency, is used to re-weight a words importance based on a frequency of a world in a document compared to the appearance in other documents [20].

4.2 Exploratory Data Analysis

To get an overall view of the data, exploratory data analysis was made on preprocessed data using graphs, word clouds and tables. Visualization was especially used to provide clarity on the characterization of various cuisines.

4.3 Classification

Various classification algorithms were used to perform the cuisine prediction based on the information from the recipes. The recipe dataset was randomly divided into training (75 %) and testing set

(25 %). The training set was used to train, while the test set was used to assess models.

4.3.1 Naive Bayes

Naive Bayes is based on applying Bayes' theorem with the naïve independence assumption between every pair of features. Gaussian naive Bayes assumes the probability of features is Gaussian. Multinomial naive Bayes implements the algorithm to the usage for text classification [21].

4.3.2 Support Vector Machine

A linear support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space using linear algebra [22].

4.4 Evaluation Metrics

To measure classification performance the following metrics were used: accuracy and F-score. Accuracy is the percentage of correct predictions. F-score is a weighted average of the precision and recall, where precision represents the ability of the classifier not to label as positive a sample that is negative and recall the ability of the classifier to find all the positive samples [23].

5. RESULTS

As an initial step, we carried out an exploratory data analysis to get a better understanding of cuisines and their characteristics. Table 2 lists average preparation time and calories per serving for each cuisine. Given the analysis, recipes from Northern Europe, Middle East and Western Europe take the longest to prepare, whereas recipes from East Asia, Latin America and Southeast Asia are generally the quickest to prepare. Furthermore, on average, Mediterranean, Oceanic and American cuisines are high in energy, on the contrary, Southeast Asian, East Asian and South Asian have recipes with lower energy values.

Table 2. Overview of the cuisines

Cuisine	Common ingredients	Average preparation time [min]	Average calories [kcal]
African	Oil, onion, lemon, clove, coriander.	51,73	399,68
Middle Eastern	Oil, onion, tomato, garlic, clove.	76,67	409,11
South Asian	Onion, oil, coriander, chili, clove.	53,74	367,50
Southeast Asian	Sauce, lime, chili, oil, sugar.	45,00	350,78
East Asian	Sauce, oil, onion, chili, rice.	40,49	363,18
Oceanic	Sugar, oil, egg.	60,36	430,70
American	Sugar, butter, oil, flour, egg.	57,68	422,37
Latin American	Onion, oil, chili, coriander, lime.	43,17	399,30
Western European	Sugar, oil, butter, egg, flour.	66,59	394,85
Northern European	Oil, sugar, onion, egg, cream.	119,59	374,61
Central European	Sugar, butter, egg, flour, oil.	62,73	402,85
Eastern European	Oil, butter, egg, flour, garlic.	57,96	390,04
Mediterranean	Oil, garlic, clove, tomato, onion.	48,68	433,36

To give an idea of the ingredients that form an integral part of each cuisine, we extracted the most common ingredients in every cuisine and visualized unigrams from the ingredient list in word clouds. As detailed in Table 2, many ingredients are frequent in all the cuisines, e.g. oil and onion, hence, these will not be useful for prediction. While others are typically used only in certain cuisines, e.g. soya sauce and clove.

Figure 1 represents word clouds consisted of the most common unigrams extracted from the ingredient list for East Asian cuisine. Although most common ingredients did not give us much insight, these word clouds do show some typical ingredients, based on which they can be distinguished from other cuisines, e.g. sugar, flour, milk, cream, chocolate, egg, mayonnaise, butter in American cuisine and soy sauce, rice, ginger, soy, chili in East Asian cuisine.

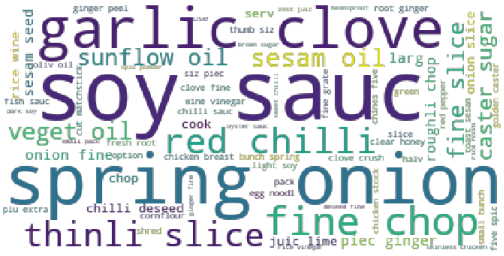


Figure 1. Word cloud for East Asian cuisine

Cuisines also differ on nutrition facts. In Figure 2, for every cuisine an average value of each nutrition per serving is presented.

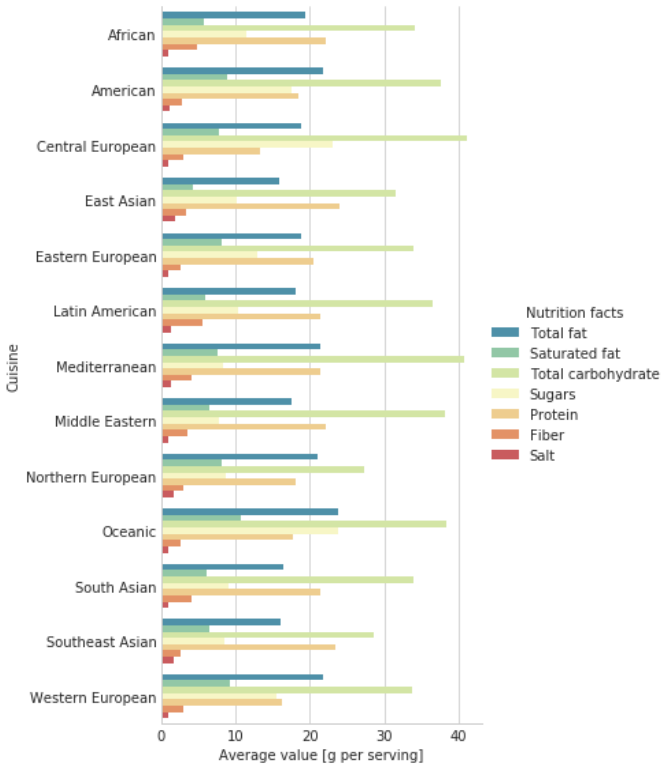


Figure 2. Nutrition facts for cuisines

In the next step, classification algorithms were applied to identify which text-based information from recipes is needed to perform

well at cuisine prediction. A classification with multinomial naive Bayes, based on the list of ingredients, proved to be the most efficient. This model yielded an accuracy of 73,8 %. Less than 1 % lower was the accuracy obtained with classification based on recipe name and more than 2 % based on preparation instructions. Classifications based on skill level, preparation time, calories and nutritional information all performed with an accuracy of about 56 %. Classification performance based on accuracy and F-score are summarized in Table 3.

Table 3. Results of classification

Information	Classifier	Accuracy	F-score
Recipe name	Multinomial naive Bayes	72,73 %	72,73 %
List of ingredients	Multinomial naive Bayes	73,83 %	73,83 %
Preparation instructions	Multinomial naive Bayes	70,97 %	70,97 %
Preparation time	Gaussian naive Bayes	55,29 %	55,29 %
	Linear SVM	55,68 %	55,68 %
Skill level	Linear SVM	56,12 %	56,12 %
Calories	Gaussian naive Bayes	55,68 %	55,68 %
	Linear SVM	55,68 %	55,68 %
Nutritional information	Gaussian naive Bayes	53,48 %	53,48 %
	Linear SVM	57,00 %	57,00 %

6. CONCLUSION

Thousands of recipes from various cuisines were analyzed with data science approaches with the objective of providing a deeper understanding of culinary cultures and cuisine prediction. While previous research efforts have mostly used only ingredients for cuisine prediction, our findings demonstrate that other text-based information extracted from recipes can be used as well. While ingredients with an obtained accuracy of almost 74 % remain to be the most efficient, cuisine prediction from recipe name and preparation instructions also performs well. Whereas prediction based on preparation time, skill level and nutrition facts were discovered to be less effective, with about 56 % accuracy.

7. REFERENCES

[1] H. Su, M. K. Shan, T. W. Lin, J. Chang, and C. T. Li, "Automatic recipe cuisine classification by ingredients," *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication - UbiComp 14 Adjunct*, pp. 565–570, 2014.

[2] S. Srinivasasubramanian, B. Kushwaha, and V. Parekh, "Identifying Cuisines From Ingredients," 2015. [Online]. Available: https://pdfs.semanticscholar.org/3daa/3c535a3c2580e69984203137db3ee6422601.pdf. Accessed on: August 16, 2018.

[3] P. Bhat, S. Gupta, and T. Nabar, "Bon Appetite: Prediction of cuisine based on Ingredients." [Online]. Available: http://cseweb.ucsd.edu/~jmcauley/cse255/reports/fa15/020.pdf. Accessed on: August 16, 2018.

[4] H. H. Holste, M. Nyayapati, and E. Wong, "What Cuisine? - A Machine Learning Strategy for Multi-label Classification of Food Recipes," 2015. [Online]. Available: http://jmcauley.ucsd.edu/cse190/projects/fa15/022.pdf. Accessed on: August 16, 2018.

[5] R. S. Verma, and H. Arora, "Cuisine Prediction/Classification based on ingredients." [Online]. Available: http://cseweb.ucsd.edu/~jmcauley/cse255/reports/fa15/028.pdf. Accessed on: August 16, 2018.R. Ghewari, and S. Raiyani, "Predicting Cuisine from Ingredients." [Online]. Available: http://cseweb.ucsd.edu/~jmcauley/cse255/reports/fa15/029.pdf. Accessed on: August 16, 2018.

[6] S. Kalajdziski, G. Radevski, I. Ivanoska, K. Trivodaliev, and B. R. Stojkoska, "Cuisine classification using recipes ingredients," *2018 41st International Convention on Information and Communication Technology, Electronics a nd Microelectronics (MIPRO)*, 2018.

[7] R. M. R. V. Kumar, M. A. Kumar, and K. P. Soman, "Cuisine Prediction based on Ingredients using Tree Boosting Algorithms," *Indian Journal of Science and Technology*, vol. 9, no. 45, Aug. 2016.

[8] T. Arffa, R. Lim, and J. Rachleff, "Learning to cook: An exploration of recipe data." [Online]. Available: https://pdfs.semanticscholar.org/3f63/269aa7910774e9386b1ffb340a9e8638c02d.pdf. Accessed on: August 16, 2018.

[9] J. Naik, and V. Polamreddi, "Cuisine Classification and Recipe Generation," 2015. [Online]. Available: https://pdfs.semanticscholar.org/aaa9/67ce597961bad308ec137a6169e1aba1fe35.pdf. Accessed on: August 16, 2018.

[10] S. Jayaraman, T. Choudhury, and P. Kumar, "Analysis of classification models based on cuisine prediction using machine learning," *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, pp. 1485–1490, 2017.

[11] H. Kicherer, M. Dittrich, L. Grebe, C. Scheible, and R. Klinger, "What you use, not what you do: Automatic classification and similarity detection of recipes," *Data & Knowledge Engineering*, 2018.

[12] T. Ozaki, X. Gao, and M. Mizutani, "Extraction of Characteristic Sets of Ingredients and Cooking Actions on Cuisine Type," *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 509–513, 2017.

[13] K. J. Kim, and C. H. Chung, "Tell Me What You Eat, and I Will Tell You Where You Come From: A Data Science Approach for Global Recipe Data on the Web," *IEEE Access*, vol. 4, pp. 8199–8211, 2016.

[14] Scikit-learn, "sklearn.preprocessing.StandardScaler." [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Accessed on: August 21, 2018.

[15] Pandas, "pandas.get_dummies." [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html. Accessed on: August 21, 2018.

[16] NLTK, "NLP with Python – Processing Raw Text." [Online]. Available: http://www.nltk.org/book/ch03.html. Accessed on: August 21, 2018.

[17] NLTK, "NLP with Python – Accessing Text Corpora and Lexical Resources." [Online]. Available: https://www.nltk.org/book/ch02.html. Accessed on: August 21, 2018.

[18] NLTK, "NLTK HOWTOs – Stemmers." [Online]. Available: http://www.nltk.org/howto/stem.html. Accessed on: August 21, 2018.

[19] Scikit-learn, "Feature extraction." [Online]. Available: http://scikit-learn.org/stable/modules/feature_extraction.html. Accessed on: August 21, 2018.

[20] Scikit-learn, "Naive Bayes." [Online]. Available: http://scikit-learn.org/stable/modules/naive_bayes.html. Accessed on: August 21, 2018.

[21] Scikit-learn, "Support Vector Machines." [Online]. Available: http://scikit-learn.org/stable/modules/svm.html. Accessed on: August 21, 2018.

[22] Scikit-learn, "Classification metrics." [Online]. Available: http://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics. Accessed on: August 21, 201

Introducing Blockchain Technology into a Real-Life Insurance Use Case

Aljaž Vodeb
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
aljaz.vodeb@student.um.si

Aljaž Tišler
Faculty of Economics and Business
University of Maribor
Maribor, Slovenia
aljaz.tisler@student.um.si

Martin Chuchurski
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
martin.chuchurski@student.um.si

Mojca Orgulan
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
mojca.orgulan@student.um.si

Tadej Rola
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
tadej.rola@student.um.si

Tea Unger
Faculty of Law
University of Maribor
Maribor, Slovenia
tea.unger@student.um.si

Žan Žnidar
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
zan.znidar@student.um.si

Muhamed Turkanović
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
muhamed.turkanovic@um.si

ABSTRACT
The paper presents an analysis of a possible introduction of the blockchain technology into an insurance business use case. The analysis is focused on the implications such an attempt can have from various standpoints and the technical workaround needed for a prototype to be implemented.

Categories and Subject Descriptors
H.3.4 [**Information Storage and Retrieval**]: Systems and Software

General Terms
Performance, Economics, Reliability, Experimentation, Security, Legal Aspects, Verification.

Keywords
Blockchain; Smart contracts; Ethereum; Insurance

1. INTRODUCTION
Blockchain technology nowadays is considered as the new IT revolution and even as the messiah for all IT-based problems. Nevertheless, as with other innovative technologies, public's hype about the technology is fading. Experts now know that the technology is useful only for specific domains and use cases as public, virtual and untrusted environment or cryptocurrency-based scenarios. Nonetheless, media is full of articles and news about corporations and companies using blockchain technology for some specific use case, which may or may not be fully meaningful. The result of such news is rising prices of cryptocurrencies and more importantly, rising stock prices of organisations [1].

The outcome of such approaches is various: (1) proposal and prototypes of blockchain-based use cases, which unnecessarily use this technology, (2) prototypes which are consistent with the technology's purpose but are unpractical and not user-friendly, and (3) failed attempts to produce a practical prototype or a production system. In this article, we explore the possibility of introducing the blockchain technology in an insurance-based use case. The aim was to explore the possible reasonableness of such a use case, its possible restrictions, limitations, advantages and disadvantages. The focus of the paper is on thus the implications of such a use case on all related processes and the overall picture of a possible implementation.

2. BLOCKCHAIN
A blockchain is an invention that can be seen as a distributed ledger of all transactions or events that have been executed and shared among distributed participants. All transactions are verified with distributed consensus inside the system. Considering basic blockchain platforms, once a transaction is recorded, it cannot be removed [2]. Group of verified transactions are stored in a block. Each block contains a cryptographic hash of the previous block and a timestamp. New linked block strengthens the integrity of the previous one, making the chain extremely tamper resistant and secure. With a public blockchain, a copy of the entire transaction database (ledger) is distributed to the network. Every person can view transactions and even participate in a consensus process.

Blockchain enables a more effective way to solve the virtual currency problem. It solves it in a distributed manner, without the need for a central authority [3]. Central authority represents costs and must be trusted to act honestly.

Public blockchain is not the only type of possible blockchain platforms. There are also private and consortium blockchains [4]. Private blockchains have write permission kept centralized to one organization. That can be useful for a single company for database management, auditing, etc. In a consortium blockchain partner companies are joined together in a trusted and adaptable network. The right to read in such blockchain types may be public or restricted to the participants.

2.1 Smart contracts

The concept of smart contract has been known since 1994, when Nick Szabo defined it as a "computerized transaction protocol that executes the terms of a contract". Inside the blockchain context, smart contracts are stored on the blockchain. They can be presented as stored procedures in relational databases. Given that smart contracts are deployed on the blockchain, they have their own unique addresses. A smart contract is invoked by executing a transaction to the unique address of the contract. It is then executed independent and automatically on each node in the network [8].
The contract has its own state and can manage assets on the ledger. It allows expressing the business logic within a programming code. A well-written smart contract should describe all the possible outcomes of the contract. This means that a function would refuse to execute in case of incorrect (inconsistent with business logic) parameters [8]. Smart contracts are deterministic - this means that the same input will always produce the same output. Implementation of smart contracts on known platforms (e.g., Ethereum), written for example in the Solidity programming language, the developer is prevented from writing non-deterministic contracts, since the programming language does not contain non-deterministic constructs. All communication with a smart contract is done through cryptographically signed transactions. This means that all blockchain stakeholders will receive a cryptographically verified trace of a contract operation.

2.2 Oracles

Smart contracts on the Ethereum blockchain platform run within the Ethereum ecosystem, where they communicate with each other. External data can only enter the blockchain (i.e. smart contracts) through external interaction using a transaction. This is also a shortcoming of the platform, because the majority of business logic is based on external data, which is thus not part of the blockchain ledger (e.g., weather, currency price) [9]. To overcome such a shortcoming an oracle can be used. Oracle is a trusted data source that sends external data to a smart contract in form of a transaction. By doing so, it relieves the smart contract of the need to directly access the desired data outside of the network. Oracles are usually offered as a third-party solution [8].

The oracle service behaves like a data courier where communication between the service and smart contract is asynchronous. First, the transaction performs the function within a smart contract in which the instructions for service are sent. The Oracle service will then obtain a result based on the parameters that will be returned to the smart contract via a special function (callback) implemented in the main smart contract in which we want data (result) from the service [9].

3. USE CASE

To test the concept of introducing the blockchain technology in a real-life business use case, we chose the insurance domain, which is also one of the promising domains for the blockchain technology [5].
A preliminary result of a market analysis has shown that a possibly meaningful, but not yet implemented use case would be the lost baggage insurance. This specific real-life use case nowadays still represents long-term problems for passengers and airlines. To make it as user-friendly and meaningful as possible, an app was envisaged. The key functionalities of such an app, as presented in Figure 1, would be: (1) user scans QR code of the flight ticket, (2) confirms read data, (3) scans barcode of baggage, (4) acknowledges terms of the smart contract, (5) info about the possible payout is provided. With help of RFID trackers at the airports the system would be able to surveillance the position of passenger's baggage based on the newly confirmed IATA resolution 753. In case of a lost or delayed baggage, an activation of a blockchain-based smart contract is executed. A compensation could be given in crypto or fiat currencies (ex. ETH, EURO), within 4 levels of payout.

Figure 1: Poster for a possible lost baggage insurance.

4. IMPLICATIONS

This section provides the implications of a possible implementation of a blockchain-based solution as presented in section 3 on three domains, legal, economic and organizational.

4.1 Legal implications

Blockchain technology as presented in section 3 raised up some legal issues. The main legal question is the General Data Protection Regulation (GDPR). GDPR is a legal framework for personal data privacy, it has been written by the European Union (EU) and became effective on May 25th. This framework is drastically changing business of any digital venture. The Regulation granted EU citizens new rights, e.g., the right to be forgotten and right to request all data storage and acquisition links. The latter allows an individual to ask an organization to delete all their personal data they store. This specific right is also the main problem in the blockchain technology. Blockchain technology relies on the principles of decentralization and immutability, which means that data stored on the ledger cannot be deleted. When this data includes personal data, we have a problem in the GDPR area. This is the main implication of this domain, since the use case worked on required the processing of personal data. The main question is thus how to process personal data with the blockchain, but still being able to delete it if needed or to process it outside the blockchain. Research shows that many experts are trying to find a solution [7]. Majority of the solutions are focused on the off/on chain paradigm, whereby personal data is never dealt with on the blockchain. Nonetheless, new problems arise as how to link off/on chain data and if the link itself is a GDPR violation.

4.2 Economic

The main goal of the solution is to enable air passengers to sign an ad hoc luggage insurance, which is tied to an airline ticket. The blockchain technology will be used for the insurance coverage and the payout of an insurance premium. The solution should allow the payment of the insurance coverage through cryptocurrencies to get the biggest customer coverage. It is a new business model, where the target group are all airline users.

The biggest negative factor associated with the possible solution is the volatility of cryptocurrencies. In practice, this represents the possibility that we lose some of our assets as a customer or as airlines. In addition to volatility, problems can occur in certain processing delays. The application itself is also linked to airline and airport data. If the system fails, automatic payment is not made possible, nor can the insurance be concluded. From an economic point of view, the application also brings many positive aspects. It is about introducing the possibility of speeding up the rigid process of current luggage insurance and redress. The cost of maintaining a blockchain network and smart contracts is not negligible. These can be covered through the annual contribution of airlines for their usage of such a possible solution. At the same time a certain percentage can be collected from each insurance.

The economic advantages of such a solution are many: (1) introduction of new technology, (2) the possibility of ad hoc insurance, and (3) a new business model.

4.3 Organizational

One of the main problems of a possible solution are of organizational structure. For it to make sense, a platform should be implemented, where all willing airlines could register and provide baggage insurance to all possible consumers. Each airline can and should have a partnership with an insurance company. Thus, to complete the registration, the airlines must provide their insurance price and max payout in case of a lost baggage. Furthermore, the solutions must be automatic and enable easy baggage check and insurance claim. A simplification of such a request comes with the IATA Resolution 753, which states that by June 2018, airline members must be able to, among others, demonstrate delivery of baggage when custody changes [6]. This furthermore implies that the ecosystem must include airports which will provide the data mentioned about the status of the baggage. Technically, a link to a web service is required, where data about the baggage is accessible.

5. PROTOTYPING

It should be emphasized that blockchain technology is a rather unexplored thing. In most cases there are no examples of good practice on process of how the introduction of the blockchain should start.

After analyzing the possible use-case and its implications we propose a prototype in a form of a decentralized application (dApp), based on the Ethereum smart contracts. The front end of the solution could be a simple Angular 2 web application with an intuitive, user-friendly interface, accessible on multiple devices. The main advantage of using a web application as opposed to device-specific applications, is the support of various operating systems and models. If a user selects to pay with cryptocurrency, he/she can use the plugin MetaMask to connect to the Web3 part of the application and send a signed transaction to a smart contract on the blockchain. According to GDPR laws, personal information needs to be delible, therefore it should be stored in a separate database off-chain, accessible through an API. Such an architecture can be given by storing airline information off-chain and non-identifying user insurance data on the blockchain.

Figure 2 presents the architecture of the possible solution. Users connect to the service through a dApp with the option to pay with crypto or fiat currencies. For clarity, the former option will be marked with the letter (a), and the latter with (b). There are two blockchains used, the Ethereum's MainNet to process payment transactions and our InsurNet for business logic (private Ethereum network). Crypto transactions are first processed on the MainNet (2a), where an oracle is triggered to convert the value into fiat (2.1a), before sending it to the InsurNet (2.2a), whereas fiat requests are processed directly through the API and if successful, forwarded towards the InsurNet (2b) to create the insurance (smart) contract. The InsurNet smart contract uses an oracle deployed at an airline to retrieve the status of the baggage (3.1 and 3.2) before processing the business logic to determine the validity of the claim. If the user is entitled to a payout, the payout oracle is called (4) to determine the correct payment method and convert currency if needed. In case the user paid in cryptocurrency (5a), the payout is processed on the MainNet (6a). Otherwise the FIAT payout is handled off-chain (5b).

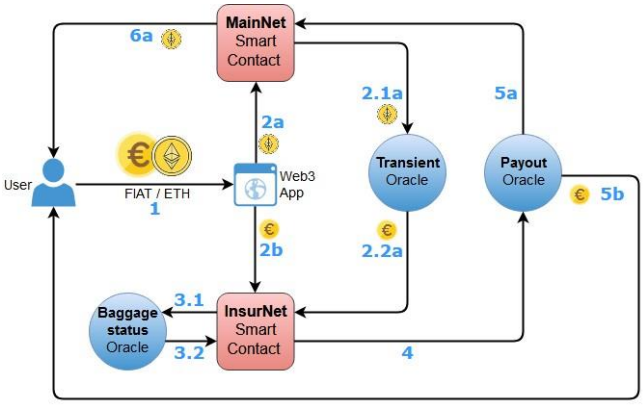


Figure 2: Architectural model of the proposed solution.

6. DISCUSSION

Due to the Ethereum Protocol, where every transaction must be validated by miners and added to the block, these can be slowly processed. When a user pays insurance with the cryptocurrency Ether into the smart contract on the MainNet and the transaction is confirmed, the function in our smart contract will trigger an event, which we can listen from outside of our dApp. We will detect the event only when the transaction is confirmed. Once our server detects the "Paid" event from the MainNet, it will create a new smart contract on our private blockchain InsurNet. This is reflected in some latency for the user. With the aforementioned oracle, we have two more. One is to verify the location of the luggage, while the other one is to process the payment when the event is triggered on InsurNet.

We can consider the following example where the user pays insurance for one luggage in the cryptocurrency. We will assume the average time to validate the transaction on MainNet is 25 seconds. The user transfers the cryptocurrency to our smart contract, where the validation of this transaction takes 25 seconds. Then, on a triggered event, oracle performs a new transaction on our network, where the transaction validation time is defined for 10 seconds. Because the user does not have the luggage yet, after three hours of landing, he performs a payout using the dApp. Transactions are done within 10 seconds. An oracle then performs a new transaction to write the current location information in the smart contract (+ 10 seconds). Since baggage is not yet available, the user is entitled to a payout, which is reflected in a new event where an oracle performs a transaction on the MainNet. The validation of this transaction takes 25 seconds. Thus, it takes at least 80 seconds for all transaction validations to complete.

7. CONCLUSION

By proposing the concept of a fully workable prototype, we demonstrate that a solution is possible. Nevertheless, after considering all the implications, we conclude that such a solution would be unpractical and not user friendly, due to all workaround needed in order to prepare a fully working technical solution. Considering the current evolutional stage of the blockchain technology, we conclude that a fully crypto-based solution can be met with approval, thus advocating the idea of the blockchain technology being seen as business disruptor in the sense of digital money.

ACKNOWLEDGMENTS

Our thanks to the public scholarship, development, disability and maintenance fund of the Republic of Slovenia and the project Following the Creative Path to Knowledge 2017 – 2020 (Po kreativni poti do znanja 2017 – 2020) - SmartInsTech.

8. REFERENCES

[1] CB Insights. Companies 'pivoting to blockchain' see huge stock spikes - but does the hype hold up? CB Insights - Research Brief. [Available] 2018. www.cbinsights.com/research/blockchain-hype- stock-trends.

[2] BlockChain Technology: Beyond Bitcoin. M. Crosby, Nachiappan, P. Pattanayak, S. Verma and V. Kalyanaraman. 2016, Applied Innovation Review .

[3] Mattila, Juri. The Blockchain Phenomenon – The Disruptive Potential of Distributed Consensus Architectures. [Available] researchgate.net/publication/313477689_The_Blockchain_Phenomenon_-_The_Disruptive_Potential_of_Distribute.

[4] EduCTX: A Blockchain-Based Higher Education Credit Platform. Muhamed Turkanović, Marko Hölbl, Kristjan Košič, Marjan Heričko, Aida Kamišalić. 2018, IEEE Access , str. 5112 - 5127.

[5] Bruno Teboul, Frédéric Maserati, Maxime Leroux. BLOCKCHAIN: CONCEPT AND APPLICATION DOMAINS. Keyrus. [Available] http://keyrus-prod.s3.amazonaws.com/Avis%20d%27expert/Blockchain/Avis% 20d%27Expert_BLOCKCHAIN-EN%20COM.pdf.

[6] IATA. Baggage Reference Manual. 2018. [Available] https://www.iata.org/publications/Documents/brm03-toc-20180523.pdf.

[7] Mercer, Rebekah. Privacy on the Blockchain: Unique Ring Signatures. arXiv. [Available] 2016. https://arxiv.org/pdf/1612.01188.pdf.

[8] Podgorelec, Blaž. Arhitektura za nadgradljivost in zamenljivost pametnih pogodb na platformi Ethereum. s.l. : DKUM, 2018.

[9] Zdun, Maximilian Wöhrer and Uwe. Design Patterns for Smart Contracts in the Ethereum Ecosystem. univie.ac.at. [Available] 8 2018. http://eprints.cs.univie.ac.at/5665/1/bare_conf.pd

A Brief Overview of Proposed Solutions to Achieve Ethereum Scalability

Blaž Podgorelec
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
blaz.podgorelec@um.si

Patrik Rek
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
patrik.rek@um.si

Tadej Rola
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
tadej.rola@student.um.si

Muhamed Turkanović
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
muhamed.turkanovic@um.si

ABSTRACT
Blockchain technology is part of Gartner's top technological trends in the following five years, whereby already moving away from the peak of the inflated expectations on its hype cycle, towards the slope of enlightenment. With the development of the blockchain technology, the emergence of completely new business processes is anticipated, as well as changes to existing business processes, which will include the use of blockchain technology in its implementation, partially or completely, thereby taking advantage of the benefits that the technology itself offers. Nevertheless, the technology has several drawbacks, whereby the most vivid is the scalability problem. With the introduction of Blockchain 2.0 and the Ethereum platform, the scalability problem seemed settled out for a moment, which proved otherwise with first generations of non-fungible tokens and high traffic. Although Ethereum is in its infancy, progress is on high tracks, with this year's focus on the infrastructure. A lot of research and work is being done on the Ethereum's layer 2 scaling solution such as the state channels, plasma and sharding. This paper presents a brief overview of the current state of the mentioned proposed solutions and some ongoing projects, which are focused on their implementation.

Categories and Subject Descriptors
H.3.4 [**Information Storage and Retrieval**]: Systems and Software

General Terms
Performance, Design, Reliability, Experimentation, Security

Keywords
Blockchain, scalability, Ethereum, channels, plasma.

1. INTRODUCTION
In recent years, on the basis of an increase in the market capitalization [1] of the Ethereum platform, the performance of which is based entirely on the blockchain technology, we can conclude that it is becoming increasingly popular. The increase in popularity consequently affects the increased number of transactions performed within the Ethereum blockchain network [2], whereby we can assume that the number of business processes that are implemented with the help of blockchain technology and Ethereum is also increasing.

All transactions transmitted on the blockchain network are irreversibly recorded in a shared ledger among all network nodes [3, 4]. Nodes in the blockchain network perform a protocol, defining the ability to create new blocks with associated transactions in an approximate 15 seconds time frame. This allows the frequency of transactions executed in the network to be approximately 7 - 15 transactions per second (tp/s) [5]. The open source Ethereum platform is based on a permisionless and publicly accessible blockchain network, which is at the same time a distributed and decentralized operating system for running smart contracts via its Ethereum Virtual Machine (EVM). Because of the platform indigenous crypto currency called Ether, generated by the blockchain network and defined by the protocol, the platform is often used as a payment system, like the Bitcoin. Therefore it is often compared to existing non-crypto payment solutions, such as Visa, which, unlike the Ethereum platform, is capable of processing a much larger number of transactions (56,000 tp/s) [6].

In the paper, we will present the problem of scaling the Ethereum network and the proposed solutions. These solutions could increase the number of transactions carried out on the Ethereum platform, thus getting closer or exceeding the processing capacity of existing non-crypto payment systems. This would enable the development and implementation of new business processes with the blockchain technology.

2. ETHEREUM SCALING PROBLEM
The current implementation of the Ethereum protocol requires the processing of all, within the network transmitted transactions, as well as the storage of all states, from each node in the network, that acts as a validator [7]. To confirm a change of the network state with a transaction, the transaction must be included in a

block created by a node, which must solve the calculation puzzle defined by the distributed consensus protocol, which is in the current Ethereum version the Proof of Work (PoW). The processing speed of the transactions is limited by the capacity of each individual node participating in the network as the transaction validator. Such an implementation of the protocol provides increased safety in terms of secure processing of transactions within the network, which is one of the key properties of such systems. At the same time, the way in which an increased security is achieved, is a major obstacle achieving a greater number of transactions carried out within the blockchain network, due to its need for heavy computation [8].

The number of transactions one block can include is limited by the number of gas (fee for processing the operations within the transaction), that can be consumed by all transactions in the block. In the future, it is possible to expect a change in the way of reaching consensus between the individual nodes in the Ethereum network. Namely, the transition to the use of the Proof of Stake (PoS) protocol is planned, which would mean that the time of block generation within the Ethereum network with associated transactions could be reduced to an average of four seconds [5]. The transition to a new protocol for reaching consensus among the nodes in the blockchain network will thus reduce the current scaling problems. In addition, the switch to PoS distributed consensus will decrease the required computational power and thus energy consumption of the network.

Changing the network consensus protocol between nodes will have a positive effect on the transaction processing frequency within the blockchain, but it is expected that the number of processed transactions will still be significantly smaller compared to the existing payment systems. Described problems in the terms of achieving greater efficiency of blockchain, assuming knowledge of its structure and understanding of the concepts of the blockchain technology, offer so-called "simple" theoretical solutions, such as:

1. It envisages the use of different "altcoins" within a variety of separate blockchain networks, which results in a strong increase in the flow rate of the performance of individual transactions within the separate blockchain networks. As a result, due to the increased number of different blockchain networks, a reduced number of nodes within different blockchain networks are expected, which would mean that separate blockchain networks will be more susceptible to attacks by malicious nodes than if all network nodes are merged within a single common blockchain network [9, 10].
2. Increasing the limit of the number of transactions per block or increasing the ceiling of fuel consumption in the case of the Ethereum protocol, theoretically implies a large number of processed transactions. Nevertheless, this requires significantly more computational power (for using the PoW protocol, or the percentage (stake) when using the PoS protocol) to validate a block with an increased number of transactions of an individual node in the network [9, 11].
3. Combining computational power (when using the PoW protocol) or stake (when using the PoS protocol) between the different blockchain networks, can theoretically increase the flow of transaction processing, but this could burden each individual node in the network due to the need for processing transactions of blockchain networks [12].

The described "simple" solutions directly relate to the so-called trilemma of blockchain technology, which says that the blockchain network can contain only two of the three features, such as:

- Decentralization
- Scalability
- Security

In the case of the use of different altcoins, this would mean increasing the efficiency (scalability) of transaction processed within the blockchain network, while in contrary a reduction of security of the network itself. The increase in the limit of number of transactions in a single block and the aggregation of computational power or the share between different blockchain networks would theoretically increase the efficiency (scalability), which would require greater use of computational power for the processing of all requirements within the blockchain network from the network nodes. This reduces the possibility of equal participation in the network by nodes with less computational power, which can lead to a reduction in the decentralization of the blockchain network by nodes who have greater computing power [8].

In the following chapters, we will present some solutions that could solve the described problem of efficiency, whereby not to affecting one of the described properties of the trilemma of the blockchain technology.

3. PROPOSED SOLUTIONS

The main concern of blockchain technology is the security and a distributed consensus in a decentralized network. The processing of every transaction by all nodes of the network is a process that provides these characteristics but does not provide enough measure for increasing efficiency and scalability. Below we describe some already proposed solutions, which can help increasing the efficiency and scalability of the Ethereum blockchain network without undermining the security and decentralization of the network as such.

3.1 State channels

One of the proposed solutions, which is currently considered to be the most mature and used, is based on the transaction processing approach outside the blockchain network (i.e. off-chain) through the establishment of state channels [13]. The proposal of the solution derives from the so-called payment channels, the purpose of which was to allow multiple micro-transactions between two users of the system without the need of transmitting each transaction through the blockchain network [14].

While payment channels focus on off-chain processing of payment transactions, the purpose of the "state channels" is to establish a channel, through which the state can be changed outside the blockchain network, between predefined participants [15]. This is because Ethereum blockchain holds the state of each defined variable of every deployed smart contract. The need to process a transaction within a blockchain network occurs only in case of disagreement about the state changed by a transaction within the established channel by any participant or in the case of a closed communication within the channel. In case that there is

no disagreement about the changed state during the communication within the established channel, this solution significantly increases the number of transactions, since it aggregates micro transactions and issues them as one in a predefined time [16].

State channels are implemented with the help of dedicated smart contracts. The establishment of communication through such a channel is carried out with a special “channel smart contract”, aimed at ensuring fair communication between participants that perform operations and record the final state into the blockchain network, after the communication has ended. In case of a conflict between participants in communication outside the blockchain (within the channel), the smart contract has the task of selecting the most relevant last state that the users still agreed on when communicating within the channel [17]. The security of such an off-chain communication approach is based on the fact that each message sent through the status channel is cryptographically signed, with the aforementioned channel smart contract having an implementation for verifying these messages. Each participant can cancel the communication at any time, and the final state that is recorded in the blockchain is that which is recognized by all participants in the off-chain communication [15].

This type of communication allows the implementation of more complex operations defined within smart contracts, completely independent of the blockchain network. Consequently this means almost instantaneous execution of operations with very low total costs of execution of all implemented channel transactions, since all transactions carried out within the established off-chain channel are aggregated into a single transaction [17, 13].

3.2 Plasma

The scalability of the Ethereum network with theoretically trillion transactions per second should be achieved by the introduction of a strategy called Plasma. Similarly, as in the solution described in Chapter 3.1, the purpose of Plasma is to implement transactions without the need for individual confirmation of each of them by the blockchain network. The solution envisages the introduction of several side chains, whereby the last state of the newly created chain being recorded in i.e. the main blockchain network. This could be implemented without any need to change the current protocol and Ethereum network. The most important factor in terms of achieving security in the Plasma solution, relates to the privilege of every user to perform transactions within any side chain (with the exception of the main Ethereum chain) and to leave the side-chain and write the final state in the main Ethereum chain - where the final valid state is defined. To prevent the recording of a false state into the main chain, the Plasma solution suggests a "Challenge mechanism", which assumes that the state that a user wants to record in the main chain is frozen for a certain period. During this period, other users can prove that the proposed state is not relevant. Because of the above mechanism, the user must provide a sum of the Ether cryptocurrency into such a transaction that writes the state into the main Ethereum chain, which if another user proves that such a transaction contains an invalid state, loses and is acquired by that user, who proved the invalid state. This mechanism could trigger a lot of false evidence of invalid transactions; therefore, a user wishing to prove an invalid transaction must pledge a sum of the Ether cryptocurrency, which in the case of false evidence of invalidity, is acquired by the user of the original transaction [18, 19].

3.3 Sharding

With the current implementation of the protocol, each node that is part of the Ethereum network must validate every transaction, which ensures a high level of network security. One solution is sharding, where the protocol would separate the network state into smaller partitions, called shards. Each shard would store its separate state and transaction history. By implementing such a protocol, certain nodes would process only the transactions of certain shards. Transactions on different shards at the same time would increase the permeability of these [20].

Sharding is a general technique used in distributed computing, the implementation of which can be expected in Ethereum by 2020 [21]. Implementation of sharding is the only one of the described scaling solutions that will practically have no impact on end users, as well as not on smart contract developers on the Ethereum platform. The system for storing states will remain the same. The change will be at layer 1 of the Ethereum Protocol. Solutions mentioned in 3.2. and 3.1. will work on layer 2 [22]. Sharding eliminates the need for the entire network (each node) to process all transactions. The result is increased number of processed transactions per second [21].

Prior to implementing sharding in the protocol, various challenges must be addressed. The main challenge is a single-shard take over attack. With such an attack, an attacker could possibly take control of the entire shard, which may result in the avoidance of sufficient validations, or even worse, to validate the blocks that are incorrect. These attacks are usually prevented by random sampling schemes. The next challenge is the availability of states between different shards. The most appropriate approach for addressing this challenge is that the effect of a transaction depends on the events that happened before in the second shard. A simple example is the transfer of money where the user A (e.g. in shard 2) transfers money to user B (e.g. in shard 7). First, a debit transaction is executed that destroys the tokens at user A (in shard 2), after which a "credit" transaction is created that creates the tokens of user B (in shard 7). This transaction has an account indicator on a "debit" transaction, which proves that the "credit" transaction is legitimate [8].

4. CONCLUSION

In the paper, we presented several different solutions, the common purpose of which is to achieve greater efficiency of scalable transaction processing in the Ethereum blockchain network. State channels move state modifications outside of the main blockchain network. The Plasma solution envisages the introduction of several blockchains, whereby each chain is used for a specific purpose. Both solutions allow users to record the final state in the main Ethereum blockchain network. We also descried the sharding solution, the introduction of which, in contrast to the above-mentioned solutions, requires the change of the lowest layer of the Ethereum protocol. All the described solutions pursue the goal of not reducing the current level of transaction processing security, as well as maintaining the decentralization of the blockchain itself in order to achieve scalability. In the future, due to the increase in the number of transactions transmitted within the Ethereum network, it is reasonable to expect several concrete implementations (Loom Network, OmiseGO, Raiden,...) of the described solutions, as well as an increased use of these in practice, since it is the increase in the efficiency of the transaction processing which is one of the key factors in achieving the

optimization of existing and new business processes, supported by the blockchain technology.

5. ACKNOWLEDGMENTS

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0057).

6. REFERENCES

[1] "Total Market Capitalization," coinmarketcap.com, 2018. [Online]. Available: https://coinmarketcap.com/charts/. [Accessed: 06-Jul-2018].

[2] "Ethereum Transaction Chart," etherscan.io, 2018. [Online]. Available: https://etherscan.io/chart/tx. [Accessed: 06-Jul-2018].

[3] B. Podgorelec, "Arhitektura za nadgradljivost in zamenljivost pametnih pogodb na platformi Ethereum," University of Maribor, 2018.

[4] M. Pustisek, A. Kos, and U. Sedlar, "Blockchain Based Autonomous Selection of Electric Vehicle Charging Station," 2016 Int. Conf. Identification, Inf. Knowl. Internet Things, pp. 217–222, 2016.

[5] F. M. Benčić and I. P. Žarko, "Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph," 2018.

[6] Visa, "Visa Inc. at a Glance," no. August, p. 1, 2015.

[7] V. Buterin, "A next-generation smart contract and decentralized application platform," Etherum, no. January, pp. 1–36, 2014.

[8] J. Ray, "On sharding blockchains," github.com/ethereum, 2018. [Online]. Available: https://github.com/ethereum/wiki/wiki/Sharding-FAQs. [Accessed: 04-Jul-2018].

[9] "The State of Scaling Ethereum – ConsenSys Media," 2018. [Online]. Available: https://media.consensys.net/the-state-of-scaling-ethereum-b4d095dbafae. [Accessed: 04-Jul-2018].

[10] A. Back, M. Corallo, and L. Dashjr, "Enabling blockchain innovations with pegged sidechains," URL http//www., pp. 1–25, 2014.

[11] GoChain, "GoChain : Blockchain at Scale," pp. 0–5, 2018.

[12] A. Judmayer, A. Zamyatin, N. Stifter, A. G. Voyiatzis, and E. Weippl, "Merged mining: Curse or cure?," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 10436 LNCS, pp. 316–333, 2017.

[13] P. Mccorry, S. Meiklejohn, and A. Miller, "Pisa : Arbitration Outsourcing for State Channels."

[14] "Lightning Network." [Online]. Available: https://lightning.network/. [Accessed: 01-Aug-2018].

[15] J. Coleman, L. Horne, and L. X. L4, "Counterfactual: Generalized State Channels," 2018.

[16] S. Dziembowski, L. Eckey, and S. Faust, "Perun : Virtual Payment Hubs over Cryptocurrencies."

[17] S. Dziembowski, S. Faust, and K. Hostáková, "Foundations of State Channel Networks," pp. 1–56, 2018.

[18] J. Poon and V. Buterin, "Plasma : Scalable Autonomous Smart Contracts Scalable Multi-Party Computation," Whitepaper, pp. 1–47, 2017.

[19] "Explained: Ethereum Plasma – Argon Group – Medium." [Online]. Available: https://medium.com/@argongroup/ethereum-plasma-explained-608720d3c60e. [Accessed: 02-Aug-2018].

[20] R. Jordan, "How to Scale Ethereum: Sharding Explained," 2018. [Online]. Available: https://medium.com/prysmatic-labs/how-to-scale-ethereum-sharding-explained-ba2e283b7fce. [Accessed: 01-Aug-2018].

[21] J. Kim, "Vitalik Buterin: Sharding and Plasma to Help Ethereum Reach 1 Million Transactions Per Second," 2018. [Online]. Available: https://cryptoslate.com/vitalik-buterin-sharding-and-plasma-to-help-ethereum-reach-1-million-transactions-per-second/. [Accessed: 01-Aug-2018].

[22] A. Rathod, "We Should See Sharding in 2020 as Part of 'Ethereum 2.0,'" 2018. [Online]. Available: https://toshitimes.com/we-should-see-sharding-in-2020-as-part-of-ethereum-2-0-eth-foundation-researcher/. [Accessed: 01-Aug-2018].

Integration Heaven of Nanoservices

Ádám Révész
EPAM Hungary
Budapest, Hungary
Adam_Revesz@epam.com

Norbert Pataki
Department of Programming Languages
and Compilers, Faculty of Informatics,
Eötvös Loránd University
Budapest, Hungary
patakino@elte.hu

ABSTRACT

Microservices have become an essential software architecture in the last few years. Nanoservices as a generalization of microservice architecture are getting more and more popular recently. However, this means that every component has more and more public interfaces, and the number of components is increasing, as well.

Integration hell had been appeared when the number of developers was increased. The developers work parallelly, so it is necessary to merge their work. Collaboration requires software support, such as version controll tools and continuous integration servers.

However, modern software development tools such as build systems, testing frameworks and continuous integration servers become sensitive regarding the version of source code to deal with. This can result in exponential explosion in many ways when nanoservices are in the focus.

In this paper, we argue for workflow that can handle this exponential explosion. This workflow can be included into continuous integration servers as jobs in order to execute test cases in a reproducible way even if the test cases deal with special environment specifications. Moreover, the workflow is able to deal with building and artifact publishing processes, as well.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement; K.6.3 [**Computing Milieux**]: Software Management

Keywords

Nanoservices, Integration, version control

1. INTRODUCTION

Microservices and nanoservices are essential software architectures recently. These software architectures have many benefits, improved scalability, separate responsibilities, better maintainability to name but a few [17]. On the other hand, having a software architecture utilizing more than 70 own built nanoservices in active development requires special care for build processes.

In terms of continuous integration (CI) and continuous delivery (CD) – modern software development process frameworks pipelines are defined as composable parts of the process describing how the product is created, transformed and delivered from making source code and configurations on developers workstations to serving them to end users [16].

The pipelines mentioned in this paper are executed by automation systems following deterministic scripts referred as "pipeline scripts".

This paper discusses the topic of bulk management of unified pipeline scripts in aspects of reproducibility, replayability, compactness and overhead of change management.

This paper is organized as follows. We present the problem of integration hell in section 2. We describe the problem in section 3. Our proposed workflow is presented in section 4. Finally, this paper concludes in section 5.

2. INTEGRATION HELL

2.1 Case

The subject of the study is a software running on top of a container orchestration system operating over multiple nodes. Using event sourcing with Command Query Responsibility Segregation (CQRS), the software utilizes over 70 services.

Every own built service is stored in its own version controller system (VCS) repository [14]. Most of them are identical in the aspect of programming language, project structure, packaging system, types of artifacts, testing frameworks, static analysis system (e.g. [11]). The discussion continues about this kind of services.

2.2 Orchestration

A container orchestration tool manages resource allocations, configurations, credentials of containers. Provides common internal network with service discovery, domain services, serving well defined endpoints for outer network communications.

In terms of scalable services, operating with nanoservices an orchestration tool must provide load balancer service over multiple nodes ensuring high availability. Also provides declarative configuration and deployment management with the ability of rolling updates and rollbacks between config-

CSS '18 Ljubljana, Slovenia

uration and deployment versions also.

Currently the industry standard for a real battle tested, serious production-grade orchestration tool is Kubernetes, developed by Google [9].

2.3 Build tools

Modern programming laguage ecosystems have their own (sometimes multiple) *package manager* for dependency handling and easy build, test, install and deploy management [12]. The common pipeline script utilizes those package managers, reaching higher level of abstraction [10]. For example:

- Java, Scala: Gradle[3], Maven[5], Ant[1]
- JavaScript - NodeJS: NPM[6], Yarn[8]
- C++: Conan [13]
- Python: Pip[7]
- Haskell: Cabal[2]
- Docker (images): Docker (registry) [15]

Closed source software projects as the subject utilize *artifact repository systems* which can serve repositories for multiple type of packages for own artifacts and serve as cache for public domain packages (in case of outage and lowering network traffic). For example: Nexus, JFrog Artifactory.

2.4 Pipelines

The services are built automatically on VCS commit on marked branches. Build pipeline scripts of actively developed services have to be in sync in order to guarantee the same level of quality and compatibility with environment (following its changes).

2.4.1 Pipeline script

A *pipeline script* is interpreted by a CI tool, a build system (e.g. Jenkins [4]), is a sequence of commands optionally separated into stages.

2.4.2 Pipeline script stage

A *pipeline script stage* is a named sequence of commands. Used for visualizing the main parts of the script, leveraging process status display during execution, variable scope segregation.

2.4.3 Pipeline command

Each *pipeline command* can be variable declaration and definition (including functions), function invocation, shell invocation.

Ideally, a *build system* has its own pipeline script domain-specific language (DSL) with an application-programming interface (API) library for common operations like VCS checkout, packaging operations, status notifications, common configuration and secret storage operations.

2.4.4 Build job

In common CI tools, each pipeline script invoked by a corresponding *build job*. These jobs contain metadata for running the pipeline script, like the location of the pipeline script itself. Storing and passing variables like job name, parameters (given on job invocation via API call or web UI).

2.4.5 Common pipeline

The subject project uses mostly Java Spring Boot nanoservices, which kind of services have a common pipeline script actively developed.

The common pipeline script contains the following stages:

- VCS checkout
- Build source code using *package manager* (like npm, Gradle, Cabal, etc.)
- Run tests on the artifact using package manager
- Sending the source code to the static analysis system
- Building Docker image artifact
- Uploading artifacts
- Announcing build status on channels (email, instant messaging)

Since these are nanoservices, their Docker images differ only on the built artifact. The configurations, including environment variables, configuration and secret files, are handled by the orchestration tool and building them into an image is an anti-pattern in this use case.

2.5 Integration hell definition

Integration hell is a place where developers have to maintain all the pipeline scripts manually for each service or use a common pipeline script and update all the source codes and configurations on each service repository to be compatible with the pipeline script. Also called *one pipeline script over all*.

3. PROBLEM STATEMENT

3.1 Build job generation

The jobs are generated depending on the VCS repository path structure. The generator job accepts the list of the service names to make build job for. The build jobs are generated from template, the only difference is in the source code repository URL and the project name.

3.2 Single pipeline script repository approach

Having dozens of services with identical pipeline scripts, it would come in hand to use the exactly same pipeline script file checked out from one build script repository.

3.2.1 Limitations of updates

The single pipeline script repository approach has multiple pitfalls. Since the the job configuration has only the repository, the branch name and the path of the pipeline script, any change on the pipeline script would affect all the build jobs at once. In this case either the ability to create experimental changes on the build scripts is lost or the ability to recreate all the build jobs without breaking any of them.

3.2.2 Lack of replayability

Other problem regarding the single repository approach is the lack of replayability. Having a case when recreating an artifact based on an older state of the service source code repository is needed, there is no guarantee the current state of the pipeline script in its repository is backward

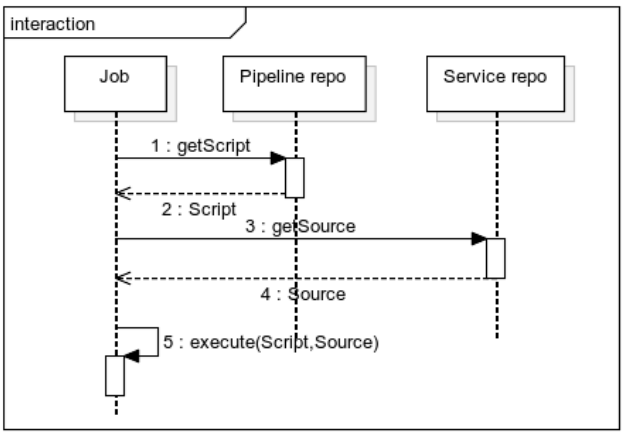


Figure 1: Sequence diagram of the proposed workflow

compatible, so there is the risk of broken or unstable build (in worse case it turns out in production). The correct build script should be searched in the history of the pipeline script repository (see Figure 1).

3.2.3 Growing overhead

The mentioned problems are getting harder to resolve as the size of the software project (the number of services) is growing. The maintenance cost of those pipeline scripts is high. Onboarding a new developer-, handing out the development of such project could be extremely difficult due to the multiple tools and sytems, scripts and their difficult dependency graph.

4. PROPOSED WORKFLOW

Addressing these problems a reasonable solution could be a property file in each service source code repository. This approach makes the generator job more difficult since every invocation it should parse the property file of every repository and generating the job according to that. An other problem is the synchronization of those property files.

4.1 Single source of truth

There is an other, more compact, more robust and more redundant way to address the problems. The *single source of truth* for service artifact build workflows should be the repository of their source code. This approach leverages the compactness of each service. The service VCS repository should contain the source code of the service, package descriptor (build scripts included) and the pipeline script. This approach can be seen on Figure 2.

4.2 Utilization of VCS

Since the VCS repository handles the pipeline script along with the source code, any arbitrary snapshot (commit) of the repository in any time of its history should contain the pipeline script which executes exactly the same pipeline with exactly the same result any time.

4.3 Keeping job generator simple

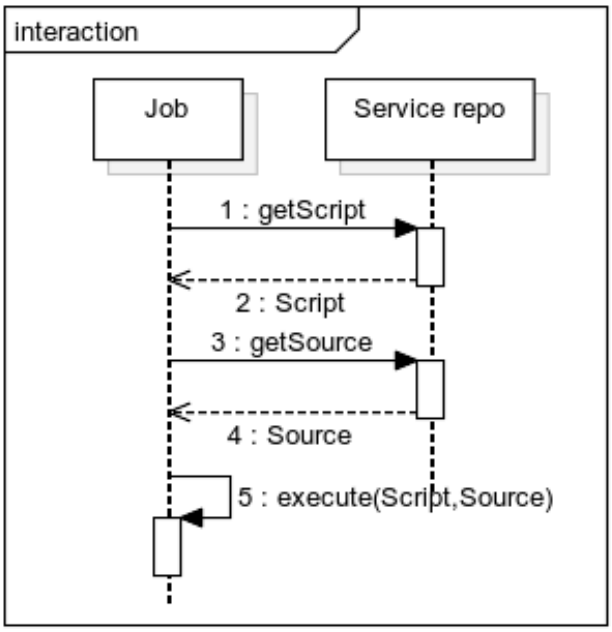


Figure 2: Sequence diagram of the single source of truth approach

This solution does not introduce the problem of difficult generator job but still carries the synchronization problem.

Pipeline scripts are being modified in multiple cases. There are cases which are not strictly drived by source code changes. Having the case of enriching the log of the pipeline script in order to leverage traceability of the process. This change is made only in the pipeline script and the side effects are present only on the pipeline script log. Has no side effect on the artifacts or test results. There are multiple open questions about which service VCS repository has to be updated first, which should be the subject of experimental changes and how to update all the other service pipeline script?

4.4 Automatized script updating

Addressing these questions, there is a pipeline script in the VCS repository but unlike the single pipeline script repository approach (see 3.2), the service build jobs are not referring to the script repository. There is a synchronization job introduced instead. The pipeline script synchronization job takes service name list as its arguments as the service build job generator job does. The pipeline script updater job has permission to update the service VCS repositories. To enforce traceability an issue id referencing an issue describing the change and its cause is recommended to be present in the commit message in all affected VCS repository. The figure 3 presents this workflow.

5. CONCLUSION

Microservices and nanoservices are popular software architectures. On the other, dealing with complex software development processes and many different development software tools, the maintenance can be a critical problem because of the combinatorical explosion.

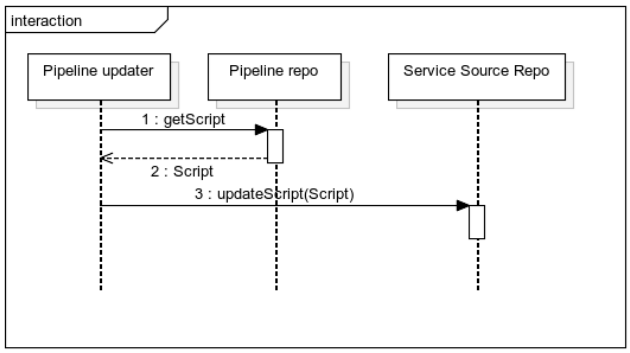


Figure 3: Sequence diagram of the proposed workflow

This solution holds some security concerns like the updater pipeline execute right has to be available for restricted group of users since the VCS enables Jenkins to commit to the master (trunk) branch.

The current prototype version is restricted to only one kind of services to upgrade their build pipeline. Enabling modular build scripts and their modular upgrade could be a next iteration. The bulk update problem could be derivated to a version controll system problem, updating common files in two or more repositories. In context of build systems like Jenkins (git) submodules could not be an optimal solution increasing complexity.

The proposed solution grants the robust script handling workflow allowing bulk pipeline script updates and replayability. It introduces some additional difficulty with the update process but it has been automatized. The approach reached a *single source of truth* state for each service artifact creation process and the refered source is the VCS repository which is a great tool to manage and observe the whole development of its content through time. The approach reduces the cost of maintaining pipeline scripts.

6. REFERENCES

[1] Ant. `https://ant.apache.org/`.
[2] Cabal. `https://www.haskell.org/cabal/`.
[3] Gradle. `https://gradle.org/`.
[4] Jenkins. `https://jenkins.io/`.
[5] Maven. `https://maven.apache.org/`.
[6] Npm. `https://npmjs.com/`.
[7] Pip. `https://pypi.org/project/pip/`.
[8] Yarn. `https://yarnpkg.com/`.
[9] D. Bernstein. Containers and cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing*, 1(3):81–84, Sept. 2014.
[10] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano. Devops. *IEEE Software*, 33(3):94–100, May 2016.
[11] G. Horváth and N. Pataki. Source language representation of function summaries in static analysis. In *Proceedings of the 11th Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems*, ICOOOLPS '16, pages 6:1–6:9, New York, NY, USA, 2016. ACM.
[12] M. P. Martinez, T. László, N. Pataki, C. Rotter, and C. Szalai. Multivendor deployment integration for future mobile networks. In A. M. Tjoa, L. Bellatreche, S. Biffl, J. van Leeuwen, and J. Wiedermann, editors, *SOFSEM 2018: Theory and Practice of Computer Science: 44th International Conference on Current Trends in Theory and Practice of Computer Science, Krems, Austria, January 29 - February 2, 2018, Proceedings*, pages 351–364, Cham, 2018. Springer International Publishing.
[13] A. Miranda and J. a. Pimentel. On the use of package managers by the C++ open-source community. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 1483–1491, New York, NY, USA, 2018. ACM.
[14] S. Phillips, J. Sillito, and R. Walker. Branching and merging: An investigation into current version control practices. In *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '11, pages 9–15, New York, NY, USA, 2011. ACM.
[15] Á. Révész and N. Pataki. Containerized A/B testing. In Z. Budimac, editor, *Proceedings of the Sixth Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications*, pages 14:1–14:8. CEUR-WS.org, 2017.
[16] S. Stolberg. Enabling agile testing through continuous integration. In *Agile Conference, 2009. AGILE '09.*, pages 369–374, New York, Aug 2009. IEEE.
[17] E. Wolff. *Microservices: Flexible Software Architectures*. CreateSpace Independent Publishing Platform, 2016.

Service Monitoring Agents for DevOps Dashboard Tool

Márk Török
Department of Programming Languages and Compilers, Faculty of Informatics, Eötvös Loránd University
Budapest, Hungary
tmark@caesar.elte.hu

Norbert Pataki
Department of Programming Languages and Compilers, Faculty of Informatics, Eötvös Loránd University
Budapest, Hungary
patakino@elte.hu

ABSTRACT

DevOps is an emerging approach that aims at the symbiosis of development, quality assurance and operations. Developers need feedback from the test executions that Continuous Integration servers support. On the other hand, developers need feedback from deployed application that is in production.

Recently, we are working on the dashboard tool which visualizes the runtime circumstances for the developers and architects. The tool requires runtime circumstances from the production environment. In this paper, we introduce our background mechanism which uses agents to retrieve runtime information and send it to our tool. We present many specific agents that we have developed for this software. Our approach deals with many useful services and tools, such as Docker and Tomcat.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging; D.2.8 [**Software Engineering**]: Metrics

Keywords

Agents, Monitoring, DevOps

1. INTRODUCTION

DevOps is an emerging approach in modern software engineering. The key achievements of DevOps are comprehensive processes from building source to deployment, continuous synchronization of development and operations in order to make every new feature delivered to the end users. DevOps emphasizes the feedback from every phase.

DevOps-culture uses a wide range of software tools. Automation of build processes is essential solution for many years. Continuous Integration (CI) servers track the version control system if a change of the source has been commited [7]. In this case, the CI server (e.g. Jenkins [1]) starts the compilation process and executes the test cases and finally, sends report to the developers regarding the changes and their effects [6]. Deployment of the compiled application and its necessary dependencies can be launched in various infrastructures [4]. Virtual machines in cloud, Docker containers on a host take part in the deployment frequently [5]. Configuration management tools (e.g. Ansible) can execute specific code snippets for the deployment. Monitoring and logging of the started application is useful to detect every kind of runtime phenomenon and orchestrate the application seamlessly [3].

However, tools landscape is missing good tools which are able to present the runtime performance of applications in staging or production environment regarding the changes of the source code. We are working on a dashboard tool to visualize how the deployed application behaves in specific environment. Many typical use-cases can be mentioned. Does the memory consumption decrease when a feature's new implementation is deployed? Which commit may cause a memory leak, if it is suspicious. Does the introduction of a new feature or API cause increase in the number of end-users? How can one compare the performance of the system if the webserver or a database server is replaced?

For our dashboard tool, we have developed many tool-specific agents to report runtime perception. Our tool visualizes the reports come from agents. We have developed agents that deal with Docker, Tomcat webserver, etc. In this paper, we present our agent-based approach and illustrate some agents' internal high-level functions.

This paper is organized as follows. In section 2, we briefly present the main concept of our tool. After, we present our agent-based approach in a detailed way with some examples in section 3. Finally, this paper is concluded in section 4.

2. DASHBOARD TOOL

A safe software development requires control over the entire software development lifecycle (SDLC). During the development, it is essential to avoid memory leakage, or overuse of the CPUs. To get a good overview of the resource utilization engineers, DevOps engineers have to keep their eyes on these units that means they have to monitor their environments by using tools that can reflect the status of the different services, databases, network I/Os, or the amount of written/read blocks.

In this chapter, we would like to give a brief introduction about our Dashboard tool which can help developers to get metrics about their environments. Developers can declare new environments on the board and assign charts to them. A chart represents a single observable unit from the real en-

CSS '18 Ljubljana, Slovenia

vironment. Metrics are provided by agents which run on the machine where the application is deployed. A continuously running agents send the gathered information back to the Listener. This way software and DevOps engineers can get an accurate picture immediately. A screenshot can be seen in Figure 1 about how a chart looks like.

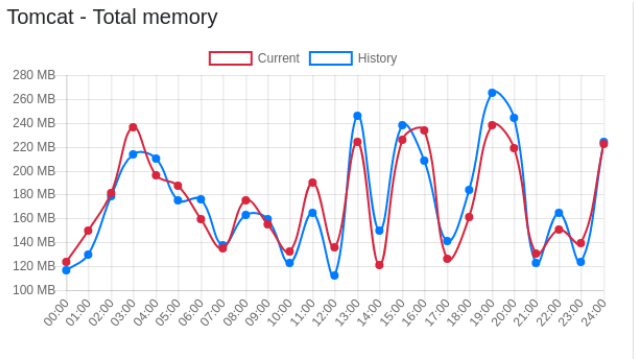


Figure 1: Memory consumption of a Tomcat instance

3. AGENTS IN OUR APPLICATION

In this section, we give a detailed view of how our agents work and what the main steps are that we kept in focus during the implementation. Before we go through the agents listed below, we would like to introduce system requirements. The target hosts are always based on Debian images, or any of its derivatives, like Ubuntu or Linux Mint. As we present later, we have strived to use as less dependencies as possible, like OS-related functionalities or commands. Most of the commands come with the basic OS, like `ps`, but some of the switches can be different on other OS, like `-eo` is Unix syntax, but using `axo` is acceptable on both Unix and BSD OS, as well.

The architecture consists of a server, the Listener and nodes which serve as hosts for the agents. In our solution, an agent is responsible for the following steps:

- After start, it runs endlessly
- Collects the information about the observed unit
- Transforms and if necessary aggregates the collected data
- Transfers the data towards the Listener server in JSON format

At first, we have to start the agent with an agent-specific sub-command and a configuration file which contains all the information that are necessary to observe the chosen unit (e.g. see Listing 1).

```
$ tomcat-agent start --file config.yml
```

Listing 1: Launching the agent

When it starts running, it validates the arguments and then parse and validates the file against the expected configuration settings that are required to the unit. Then it starts monitoring and collecting metrics in a specified time period. Beside these steps, an agent also has minor characteristics, like

- Runs as a daemon
- Validates the configuration file to have proper keys
- Validates the values in the configuration yml file
- Checks whether the related OS-level dependencies exist
- Transfers the collected metric to JSON

Beside these steps, an agent also has minor characteristics. It runs as a daemon. It checks whether the related OS-level dependencies exist. It transfers the collected metric in JSON.

All agents require a file that contains specific information for the observed unit, as well as, parameter for the connection to the Listener. One file can be used by many agents, and one file can contain configurations for multiple observed units.

Here we detail some of the agents mechanism, how they work and what information we can get from the unit.

3.1 Tomcat

Tomcat is one of the most popular and widely-used application server among Java developers. It provides a simple dashboard-like landing page where software and DevOps engineers can manage the deployed packages. Via this page those users, who are dedicated to enter the server, can check the state of their applications. This can be a simple healthcheck, the number of threads or how much memory is available for Tomcat to allocate more space for the applications.

The tomcat agent monitors both the inside status page and the process itself as well. In the configuration file (see Listing 2), DevOps engineer has to declare specific parameters.

```
uri: 'localhost'
port: 8080
username: 'admin'
password: 'admin'
pid: 24567
```

Listing 2: Agent configuration file example

If `pid` is not available, agent monitors the inside status page only. An example metric that the agent is intended to send towards the Listener can be seen in Listing 3.

```
{
  "status": {
    "jvm": {
      "memory": {
        "free": 2335645,
        "total": 88234123,
        "max": 2453422
      }
    },
    "connector": {
      "requestInfo": {
        "maxTime": 12,
        ...
      },
      "threadInfo": {
```

```
        "maxThreads": 1,
        "currentThreadCount": 1,
        "currentThreadBusy": 0
      }
    }
  }
}
```

Listing 3: Example metric sent by the tomcat agent

3.2 Docker

Containerization is new directive in virtualization: this lightweight approach supports operating system-based isolation among different pieces of the application. Containerization is on the crest of a wave since Docker has been developed. Docker provides a systematic way to automate the fast deployment of Linux applications inside portable containers [2].

The name of docker is basically almost equivalent of *container* for most of the engineers. Docker, just like Tomcat, provides a calculation on how much memory it consumes or what the total bytes of the received and transmitted data is over the network for each container. These are the *stats*. Without declaring any specific container name in the config file, the agent sends information about all the containers at the same time that are shown up in the stats. An example message can be seen in Listing 4.

```
{
  "containers" : [
    {
      "pid": 38,
      "name" : 'jingle_bell',
      "cpu" : 1.86,
      "mem": {
        "usage": "168.2M",
        "limit": "15.43G",
        "percentage": 1.06
      },
      ...
    }
  ]
}
```

Listing 4: JSON message example sent by agent

3.3 Log

One of the most important mirror of the status of an application is its logs. It could contain all the steps that an execution takes and provide those steps in different granularity.

The two main approaches in case of this agent are, first, get the last n messages from the log and forward it to the Listener, and second, get the number of the different severity levels. The earlier can provide a view of the latest messages, which is a talkative information based on the error or exception messages raised in the code. The latter one can show the ratio of the different levels giving a clear overview how much warnings or errors get hit during the execution. To get these two metrics we mentioned above, engineers have to use such a configuration seen in Listing 5.

```
...
path: '/logs'
file: 'observed'
format: 'SEVERITY||'
number_of_lines: 10
```

Listing 5: Example configuration for the log agent

The `path` tag is responsible for the path of the folder which is considered as a log folder and `file` is the observed unit. To distinguish an *ERROR* leveled message from other messages that contains the word *error*, engineers have to declare the format of the log. The last key is responsible for the number fetched and forwarded messages. A sent message example sent can be seen in Listing 6.

```
{
  "lines": ["..."],
  "severity": {
    "info": 655,
    "warning": 848,
    "error": 2,
    "fatal": 0
  }
}
```

Listing 6: JSON message example sent by the log collection agent

Since an agent is run on a machine by an arbitrary user, the software, DevOps and test engineers have to take care that the observed log can be any file depends on the privileges of the user.

3.4 Host Machine

The host machine which the agent is executed on, can be a real machine, a virtual machine or a container whether it is on local or on remote. Whichever the host machine is, from the agent perspective they are the same. From inside out it seems that machine has memory, CPU (or GPU), hard disk and other resources. These resources are reachable for the agents that means agents can use them. Having a picture about the usage and consumption of these resources are essential.

With this agent, we can monitor the above-mentioned resources and gather their metrics. These metrics are cumulated, agent takes, for example the total memory, the total swap memory or the size of the available space on the hard disk, regardless which processes use them.

Here we would like to give a view which metrics are taken during the agent's execution. We arranged the resources into three groups. All the metrics belong to the *memory*, or *CPU*, or *disk storage* (volume).

3.4.1 Memory

Memory has multiple parts from total to used to swap. To get an accurate picture about the consumption we use, multiple commands that can help calculating the usage of the different parts. The agent uses `free` (see Listing 7), `/proc/meminfo` and the `vmstat` commands to get metrics about the memory (see Listing 8). All of them provide information about how much total memory is in that host, what the size of the cached swap or how much memory is free or how much is available for allocating new processes.

```
$ free -m
```

```
      total  used  free ...
Mem:  15802  5485  5707 ...
Swap:  2047     0  2047
```

Listing 7: Using the `free` command

```
{
  "Mem": {
    "total": 15802,
    "used": 5485,
    "free": 5707,
    "shared": 2088,
    "buff/cache": 4609,
    "available": 7894
  },
  "Swap": {
    "total": 2047,
    "used": 0,
    "free": 2047
  }
}
```

Listing 8: Sent message about memory consumption

3.4.2 CPU

There are plenty of tools that provide the opportunity to monitor the usage of the CPU. Some of them are part of the default OS, then the rest come as a third-party tool and require installation with privileges. We took the focus on those tools that are part of the OS, or used in wide range, like `vmstat`, or `iostat` (see Listing 9). Both tools can provide a picture of the CPU utilization in percentage.

```
$iostat -c
Linux 4.15.0-32-generic 2018-08-25 _x86_64_ (8 CPU)

avg-cpu: %user %nice %system %iowait %steal %idle
         24,97  0,03    6,07    0,03   0,00 68,90
```

Listing 9: Using the `iostat` command

The agent sends the above information towards the Listener as it seen in Listing 10.

```
{
  "user": 24.97,
  "nice": 0.03,
  "system": 6.07,
  "iowait": 0.03,
  "steal": 0.00,
  "idle": 68.9
}
```

Listing 10: Sent JSON message about CPU usage

3.4.3 Volume

Volume usage does not belong to the major metrics of the previously mentioned three units. Though it can tell useful information about a running application. To get a metric about the volume agent uses `df` (see Listing 11) and `du` commands. Both of them are responsible for giving a view of how much space is taken by a folder or how the size of the local storage changes. Moreover, agent can be parameterized. It takes the path to the observed folder or partition of the storage of type of the disk. The agent sends aggregated information as it seen in Listing 12.

```
$ df -t ext4
Filesystem     1K-blocks      Used Available Use% Mounted on
/dev/nvme0n1p5 120462064 77259492  37040396  68%  /
```

Listing 11: Using the `df` command

```
{
  "filesystem": "/dev/nvme0n1p5",
  "1k_blocks": 120462064,
  "used": 77259492,
  "available": 37040396,
  "use": 68,
  "mounted_on": "/"
}
```

Listing 12: Sent JSON message about volume usage

4. CONCLUSION

DevOps is an emerging approach that aims at the symbiosis of development, quality assurance and operations. Developers need feedback from the test executions that CI servers support. On the other hand, no tools have been created that support feedback from the production enviroment to the developers to follow up the code changes and its effect on the end-users and the production or the staging environment.

In this paper, we argue for a new tools into the DevOps toolset. The aim of this tool is retriving and visualizing the runtime circumstances of deployed application because this information can be essential for the developers and architects. For this tool, we have developed many agents to collect the runtime performance information from specific services. In this paper, we presented the mechanism of some specific agents in Linux environment.

5. REFERENCES

[1] Jenkins. `https://jenkins.io/`.

[2] D. Bernstein. Containers and cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing*, 1(3):81–84, Sept. 2014.

[3] P. P. I. Langi, Widyawan, W. Najib, and T. B. Aji. An evaluation of twitter river and logstash performances as elasticsearch inputs for social media analysis of twitter. In *Information Communication Technology and Systems (ICTS), 2015 International Conference on*, pages 181–186, New York, Sept 2015. IEEE.

[4] M. Leppänen, S. Mäkinen, M. Pagels, V. P. Eloranta, J. Itkonen, M. V. Mäntylä, and T. Männistö. The highways and country roads to continuous deployment. *IEEE Software*, 32(2):64–72, Mar 2015.

[5] Á. Révész and N. Pataki. Containerized A/B testing. In Z. Budimac, editor, *Proceedings of the Sixth Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications*, pages 14:1–14:8. CEUR-WS.org, 2017.

[6] J. Roche. Adopting DevOps practices in quality assurance. *Commun. ACM*, 56(11):38–43, Nov. 2013.

[7] S. Stolberg. Enabling agile testing through continuous integration. In *Agile Conference, 2009. AGILE '09.*, pages 369–374, New York, Aug 2009. IEEE.

Incremental Parsing of Large Legacy C/C++ Software

Anett Fekete, Máté Cserép
Eötvös Loránd University
Faculty of Informatics
Budapest, Hungary
{hutche, mcserep}@inf.elte.hu

ABSTRACT
CodeCompass is an open source project intended to support code comprehension by providing textual information, source code metrics, version control information and visualization views of the file and directory level relations for the analyzed project. Regarding the typical software development methodologies (especially the agile ones), only a smaller portion of the code base is affected by any change during a shorter amount of time (e.g. between nightly builds), therefore parsing the entire project each time is unnecessary and expensive. A newly introduced feature, incremental parsing is intended to solve this problem by only processing files that have been recently changed and leaving the rest alone. This is achieved by the maintenance of the project workspace database followed by the partial parsing of the project. The feature has been tested both on medium and large scale projects and proved to be an effective tool in CodeCompass.

Categories and Subject Descriptors
D.2.3 [**Software Engineering**]: Coding Tools and Techniques; D.3.4 [**Programming Languages**]: Processors

General Terms
Management, Languages

Keywords
code comprehension, software maintenance, static analysis, incremental parsing, C/C++ programming language

1. INTRODUCTION
One of the main tasks of a code comprehension software tool is to provide exact textual information and visualization views regarding the analyzed codebase to support the (newcomer) developers in understanding the source code. For an enterprise software under development this requires the frequent static reanalysis of the program, which could take several hours for a large legacy software.

Performing a complete static analysis each time is a significant waste of computational resources, since in most cases (e.g. between nightly builds) only a few percent of the file set has been affected by any change. In order to boost the parsing and compilation process and to provide richer user experience in integrated development environments (IDEs) [8], the concept of incremental parsing and compilation has been researched since decades. More recently further approaches, like the involvment of version control systems into incremental parsing [14] and the lazy analysis [10] have been studied. A great overview of pratical algorithms and the exsiting methodology is given by Tim A. Wagner in [13]. C/C++ language-specific compilation tools [12, 4] and programming environments [7] supporting incremental parsing have also emerged as an advancement.

CodeCompass [9] is an open source, scalable code comprehension tool developed by Ericsson Ltd. and the Eötvös Loránd University, Budapest to help understanding large legacy software systems. Its web user interface provides rich textual search and navigation functionalities and also a wide range of rule-based visualization features [5, 6]. The code comprehension capabilities of CodeCompass is not restricted to the existing code base, but important architectural information are also gained from the build system by processing the compilation database of the project [11]. The C/C++ static analyzer component is based on the LLVM/Clang parser [1] and stores the position and type information of specific AST nodes in the project workspace database together with further information collected during the parsing process (e.g. the relations between files). By introducing the concept of *incremental parsing* into CodeCompass we can detect the added, deleted or modified files in the program and carry out maintenance operations for the database of the code comprehension tool in only the required cases. Thus the required time of the reanalysis can be reduced by multiple magnitudes.

In this paper first we present our research in Section 2 on how we extended the static analytical capabilities of the CodeCompass code comprehension tool with incremental parsing. Then Section 3 demonstrates the usability of the concept by showcasing incremental parsing and measuring its performance on a medium and a large size C/C++ software. Finally, Section 4 concludes the results and discusses further research opportunities.

2. METHODOLOGY
A major consideration of the introduced incremental parsing feature was to integrate it seamlessly into the existing parsing process by not differentiating in how an initial or a follow-up incremental parse should be initiated. This was achieved by utilizing the *partial parsing* feature of CodeCompass, which means that the tool is capable of continuing a previously aborted analysis, by omitting the already parsed files which are present in workspace database.

Therefore the main concept of the introduced incremental parsing feature consists of two steps: *i*) perform a *database maintenance* operation, where the project workspace is restored into a state that *ii*) the existing *partial parsing* can finish the procedure.

2.1 Determining file states

When a new parse is being done in incremental mode, the state of each file is determined first. Let F_{DB} be the file set stored in the workspace database and F_{DISK} be the file set stored on the disk. An $f \in F_{DB} \cup F_{DISK}$ file may take one of the three states listed as follows.

Added files f is added to the project since the latest parse if $f \in F_{DISK}$ but $f \notin F_{DB}$.

Deleted files f is deleted from the project if $f \in F_{DB}$ but $f \notin F_{DISK}$.

Modified files f is modified when $f \in F_{DB} \cap F_{DISK}$ at the time of the new parse but its content has changed since the latest. This can be determined by comparing the contents that are stored in the database and on the disk, or by their respective hashes for performance optimization.

2.2 Header inclusion traversal

Specifically when parsing a C or C++ language project, changes in header inclusions provide one more challenge to tackle. Upon the modification of a header file all further files in the inclusion chain depending on it should be considered as modified, even without containing any direct changes themselves. Therefore when determining the *modified* state of a file as defined in Section 2.1, the set of files defined by the header inclusion relationships transitively should be checked for changes. There are two approaches for this, as described below and shown in Figure 1.

Definition 1. For files a, b and c, given that a is included by b and b is included by c, we say that file a is in an upward connection with b and accordingly file c is in a downward connection with b.

Upward traversal model The upward traversal model depends on the upward connection between files. When resolving the state of file a, its included headers have to be checked for modifications transitively.

Downward traversal model Similarly, the downward traversal model uses the downward connections that can be found between files. If a file a is resolved as modified, all files that include a can be marked as modified transitively. Note that with this method, the state of any marked files can be considered final and can be omitted from further inspections.

THEOREM 1. *The downward traversal model has better computational complexity over the upward traversal model, and therefore is preferred to be used through the incremental parsing.*

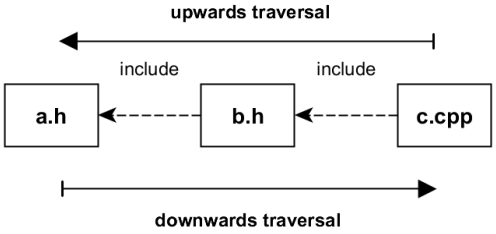


Figure 1: Traversal directions

PROOF. Let $G = (V, W, E)$ be the directed acyclic graph (DAG) of header inclusions with V containing the file set as vertices and E being the set of upward connections, $n := |V|$, $e := |E|$. Let $W \subseteq V$ denote the set of directly changed files, $k := |W|$.

Let $N_G(v)$ be the neighborhood file set of vertex v in G, so $w \in N_G(v) \Leftrightarrow (v, w) \in E$. Therefore for a file v we can define the directly included file set as $N_G(v)$ and the includer files of v as $N_{G^T}(v)$, where G^T is the transpose graph of G.

We define $up(G, v)$ and $down(G, v)$ as the file set result of the upward and downward traversal for $v \in V$ in G by the corresponding traversal model, as formally described below:

$$up(G, v) = \{v\} \cup \forall_{w \in N_G(v)} : up(G, w) \tag{1}$$

$$down(G, v) = \{v\} \cup \forall_{w \in N_{G^T}(v)} : down(G, w) \tag{2}$$

As a simplification in our model lets assume a uniform distribution of header inclusions among the files. Since $\sum_{v \in V} deg^+(v) = \sum_{v \in V} deg^-(v) = e$, the average in-degree and out-degree for a file v is $deg^+(v) = deg^-(v) = \frac{e}{n}$, which will be denoted with d henceforth. As a consequence the length of the longest path in G is $log_d n$, which is the length of the longest header inclusion chain in the project, since G was defined as a DAG.

Therefore the asymptotic tight bound both for $up(G, v)$ and $down(G, v)$ can be calculated as:

$$\Theta(up(G, v)) = \Theta(down(G, v)) = d^{log_d n} = n \tag{3}$$

We define $up(G)$ and $down(G)$ as the upward and downward traversal algorithms which determines indirectly changed files in V through header inclusions from W by the corresponding traversal model. We define the computational complexity of the algorithms as the number of files checked for changes in their content (or by their hash). Based on Equation 3, the asymptotic tight bound both for $up(G)$ and $down(G)$ can be calculated as:

$$\Theta(up(G)) = \sum_{v \in V} \Theta(up(G, v)) = n^2 \tag{4}$$

$$\Theta(down(G)) = \sum_{w \in W} \Theta(down(G, w)) = k * n \tag{5}$$

Since $k \leq n$ and in a typical use case for incremental parsing $k \ll n$: $\Theta(down(G)) < \Theta(up(G))$. □

An example for the downward traversal model is showcased in Figure 2. On the left side of the figure the example file set is shown with header inclusion dependencies denoted as arrows between them. Directly modified files are marked with a dark background, while files requiring expansion through traversal to find indirectly changed files are marked with an italic font. Note, that these two categories are equivalent in the initial stage. On the right side of the figure the effects of downward traversing `a.h` is demonstrated: files `c.h`, `d.h`, `f.cpp` and `g.cpp` are also detected as indirectly changed files. While `c.h` was also a directly modified file, observe that it no longer requires downward traversal.

2.3 Database maintenance

As mentioned above, incremental parsing includes some maintenance of the existing database depending on the state of changed files.

1. *Added files* are perceived as new files to the project and therefore are registered into the database.
2. *Deleted files* need to be purged from the database as they have been removed from the project.
3. *Modified files* are handled as if they were a combination of deleted and added files. First, they are completely wiped out from the database – meaning that all their AST related information and file level relations are erased –, thus considering them deleted, then re-registered like newly added files. Directory level relations are not sufficiently maintainable, but these relations can be effectively computed runtime, on-demand from the file level relations.

3. EXPERIMENTAL RESULTS

The go-to projects on which CodeCompass is usually tested are the Xerces-C++ [3] and LLVM [2] projects. Both are open source projects that have been under development for several years and therefore are considered legacy projects. Incremental parsing was also tested on these two as Xerces-C++ is a medium size and LLVM is a large-scale project and contain enough files (respectively 347 and 2845) to produce a significant difference in runtime between even small portions of changes in the number of files.

Incremental parsing is aimed to reduce the parsing time of builds, especially nightly builds, therefore it was tested on 1, 5 and 10 percent change of the file set, since no bigger difference between two builds is presumable. The changeset was generated automatically by random selection of files.[1] Table 1 shows the results for Xerces-C++, while Table 2 and Table 3 depict the results for LLVM. All measurements were carried out on a standard notebook computer, parsing on 2 processor cores.

In order to keep database consistency in case of a graceful abort or unexpected termination of the parser module, the basic concept is that the maintenance operation of incremental parsing must be performed in a transactional mode, in one of the following ways:

[1]Only leaf nodes from graph G introduced in Section 2.2 were included in the changeset, so header inclusions did not affect the number of changed files.

Table 1: Time measures for incremental parsing the Xerces-C++ project

Parse type	Changed files	Time
Full parse	–	2 min 49 sec
1% change	3	10 sec
5% change	17	21 sec
10% change	35	49 sec

Table 2: Time measures for incremental parsing the LLVM project by one atomic transaction

Parse type	Changed files	Time
Full parse	–	5 h 46 min
1% change	28	7 min 30 sec
5% change	142	1 h 58 min
10% change	284	2 h 45 min

- Carry out all deletions from the database in one *single transaction*, so the maintenance is either completely executed, otherwise no changes are performed.
- Generate multiple *file level transactions*, so information regarding a file is either cleaned from the database or the file is untouched, therefore a consistent state of the database is always kept.

Table 2 and Table 3 compare the differences when the database maintenance is executed through a single and by file level transactions. It is clear that the extensive size of the database rollback log containing all the deletion operations for a larger quantity of files can significantly hinder the effectiveness of incremental parsing, providing significant difference in the timespan of incremental parsing for large size projects like LLVM. Hence while a single transaction may provide stronger guarantees, file level transaction proved to be a more adequate solution, where the required time is more or less linear with the quantity of parsed files, depending on the length and content of the files in question.

Table 3: Time measures of incremental parsing the LLVM project by file level transactions

Parse type	Changed files	Time
1% change	28	9 min 30 sec
5% change	142	49 min
10% change	284	1 h 21 min

4. CONCLUSIONS

Incremental parsing was introduced into CodeCompass to reduce the costs of parsing, both time and computational resources, by omitting unchanged files in the project. The feature distinguishes added, deleted and modified files and handles them accordingly. The early tests of incremental parsing were run on the Xerces-C++ and LLVM projects and showed that it works according to its original purpose, especially in decreasing the timespan of parsing. While the results are promising, further challenges include the improved reduction of the timespan required by incremental parsing through parallelizing the process.

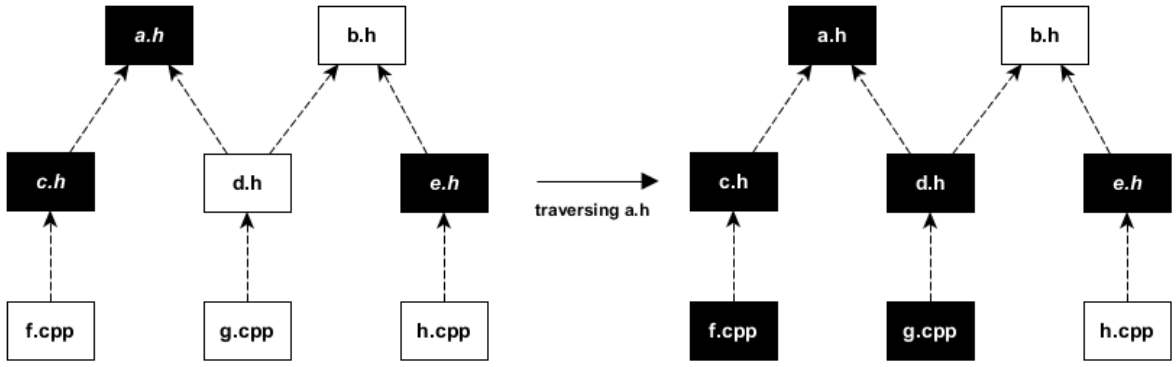


Figure 2: Downward traversing of *a.h* demonstrated on a showcase file set.

5. ACKNOWLEDGMENTS

This work is supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

6. REFERENCES

[1] Clang: a C language family frontend for LLVM. `https://clang.llvm.org/`.

[2] The LLVM Compiler Infrastructure. `https://llvm.org/`.

[3] Xerces-C++ XML Parser. `https://xerces.apache.org/xerces-c/`.

[4] Zapcc – A (Much) Faster C++ Compiler. `https://www.zapcc.com/`.

[5] T. Brunner and M. Cserép. Rule based graph visualization for software systems. In *Proceedings of the 9th International Conference on Applied Informatics*, pages 121–130, 2014.

[6] M. Cserép and D. Krupp. Visualization Techniques of Components for Large Legacy C/C++ software. *Studia Universitatis Babes-Bolyai, Informatica*, 59:59–74, 2014.

[7] M. Karasick. The Architecture of Montana: An Open and Extensible Programming Environment with an Incremental C++ Compiler. *SIGSOFT Softw. Eng. Notes*, 23(6):131–142, Nov. 1998.

[8] R. Medina-Mora and P. H. Feiler. An incremental programming environment. *IEEE Transactions on Software Engineering*, (5):472–482, 1981.

[9] Z. Porkoláb, T. Brunner, D. Krupp, and M. Csordás. Codecompass: An open software comprehension framework for industrial usage. In *Proceedings of the 26th Conference on Program Comprehension*, ICPC '18, pages 361–369, New York, NY, USA, 2018. ACM.

[10] V. Savitskii and D. Sidorov. Fast analysis of source code in C and C++. *Programming and Computer Software*, 39(1):49–55, 2013.

[11] R. Szalay, Z. Porkoláb, and D. Krupp. Towards better symbol resolution for C/C++ programs: A cluster-based solution. In *IEEE 17th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 101–110. IEEE, 2017.

[12] T. Tromey. Incremental compilation for GCC. In *Proceedings of the GCC Developers' Summit*. Citeseer, 2008.

[13] T. A. Wagner. *Practical algorithms for incremental software development environments*. PhD thesis, Citeseer, 1997.

[14] T. A. Wagner and S. L. Graham. Efficient and flexible incremental parsing. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 20(5):980–1013, 1998.

Visualising Compiler-generated Special Member Functions of C++ Types

Richárd Szalay
Eötvös Loránd University, Faculty of Informatics
Department of Programming Languages and Compilers
Budapest, Hungary
szalayrichard@inf.elte.hu

Zoltán Porkoláb
Eötvös Loránd University, Faculty of Informatics
Department of Programming Languages and Compilers
Budapest, Hungary
gsd@elte.hu

ABSTRACT

In the C++ programming language, special member functions are either user-defined or automatically generated by the compiler. The detailed rules for when and how these methods are generated are complex and many times surprise developers. As generated functions never appear in the source code it is challenging to comprehend them. For a better understanding of the details under the hood, we provide a visualisation method which presents generated special functions in the form of C++ source code that in effect identical to their implicit versions.

CCS CONCEPTS

• **Software and its engineering** → **Source code generation**; **Software maintenance tools**; • **Human-centered computing** → *Information visualization*;

GENERAL TERMS

programming languages, software development, visualisation

KEYWORDS

C++ programming language, compilers, code comprehension, code design

1 MOTIVATION

Languages supporting the Object-oriented programming (OOP) paradigm define a central principle of object *lifetime* which is surrounded by construction/initialisation and destruction/finalisation. In the **Java** programming language, apart from the basic default construction – where everything is initialised to the respective zero value – the developer must explicitly state their intent for different construction logic, custom finalisation. A special case is when a new object is created from an already existing one, where deep copy (*clone*) operations or conversions might be warranted. In **C++**, however, the Language Standard specifies that these aforementioned actions, in the form of *special member functions* [8], should have a default implementation automatically generated by the compiler if the user does not explicitly write them. The rules which dictate the conditions for generating the special member functions and their behaviour can appear dauntingly complex, and subsequent versions of the language standard may revise and elaborate these rules, increasing their complexity. The most recent, and most significant such change was with the release of the *C++11* standard, which introduced *move semantics* [9].

Modernising code initially written for an older standard can be cumbersome as the behaviour of special members are never directly expressed, yet relied upon by the most trivial codes. What's more, the compiler is free to lazily evaluate the generation of these members, which results in one such member's non-availability to only be reported when its usage was attempted. In case the used software library is outdated or not easily modifiable, or not open source, this can result in loss of run-time performance or development effort wasted on having to redesign parts of the software. For discovery and understanding of the existence and behaviour of these methods, developers can either consult the Language Standard, read Abstract Syntax Trees (ASTs), or view the disassembly of the binary — none of which is favourable for the average developer.

```
1  #include <iostream>
2  struct A { int x };
3  int main() {
4    A a1;      // <- Default constructor called.
5    a1.x = 5;
6    A a2(a1); // <- Copy constructor called.
7    a1.x = 6;
8
9    // Will print "6 5".
10   std::cout <<  a1.x << "␣" << a2.x;
11 }
```

Listing 1: Example code which uses a default and a copy constructor.

To aid ongoing development and code comprehension of projects we introduced a tool that allows pretty-printing the visual representation of special member functions that is the closest to how they would be written by developers. To further this aid, we don't only show the compiler-generated special members, but provide a subset of the type's all member functions which shows both user-written – e.g. a constructor that initialises from a different data type – and the standard, implicit ones. We used the open source LLVM/Clang Compiler Infrastructure [16] for parsing and generation.

The rest of the paper is organised as follows. In Section 2 we discuss the purpose and rules of C++ special member functions. Then, Section 3 describes the implementation approach and challenges faced with respect to pretty-printing and presentation to the developers. The paper concludes in Section 4.

2 C++ SPECIAL MEMBER FUNCTIONS

Special member functions in C++ denote the functions that are necessary for the management of instances' lifetime. [12] These are the *constructors*, the *assignment operator* and the *destructor*.

2.1 Constructors

Constructors are responsible for the initialisation of an object. They are usually executed together with the memory allocation for the instance. Unless the user specifies and provides any constructor function, both C++ and Java will generate a *default constructor*. In Java, this function initialises every data member to their respective zero value, such as integer 0, rational 0.0, the `\0` character, or a `null` reference. In C++, the initial state of the members depend on the storage scope of the object – in most cases, the memory garbage is retained from the memory block where the object is allocated. Unlike Java, however, the default constructor is not created if at least one data member does not have a default constructor.

Another case of construction is when a new object is initialised from the state of another, already living object of the same type. In Java, this functionality can be achieved in multiple ways, one of which is by using the special `clone()` function. This function is defined in `Object`, and performs a *shallow copy* of the instance in question, only initialising the new object's members to the same value of the cloned one [4, 11]. In case of references to other objects results in *aliasing*, the sharing of the same resource – usually an internal buffer – by two separate entities. Another problem with `clone()` is that the existence of the cloneability marker and the respective method must exist through the whole chain of the type hierarchy – it is usually referred to as an epidemic [10]. What's more, cloning does not actually invoke a construction, but rather creates a copy of the memory's snapshot, which means that business logic strictly bound to a constructor, such as initialisation of read-only members, cannot be done. In C++, the default behaviour of the copy constructor is to run the copy construction of every data member. For fundamental types, this means a copy of the value, and for more complex types their respective copy constructors are called. Thus, in case a custom resource which can be properly deep-copied is used the copy constructor that is generated for the object using this resource will be sufficient.

2.2 Destructor

The destructor or finalise is called at the end of an instance's lifetime and is responsible for tearing down the state of the instance. This most commonly means releasing resources, performing clean-up tasks and committing changes, e.g. to a database. In Java, the `finalize()` method's implementation is run for an object at an unspecified point in time when the runtime's garbage collector decides that the object is to be reaped. [3] The behaviour differences between Java Virtual Machine versions and the general looming of a finalisation never happening for an instance resulted in a consensus on not using `finalize()` – it has also been deprecated since Java 9. Instead, the *AutoCloseable* design pattern is used that explicitly requires writing a `close()` method which executes teardown logic, but can be called arbitrarily by the developers when teardown is deemed necessary, such as at the end of finishing a database operation. In C++, a destructor can be written by the user or is automatically generated by the compiler. It is always executed immediately when an instance's lifetime ends. The generated destructor does nothing in its body, and then the destructor of each data member is executed individually – as their lifetimes also expired. Thus an implicit destructor always exists unless a data member's destructor is explicitly hidden – this is a common practice for scenarios where a controller has to ensure an orderly or batch destruction.

2.3 Assignment operators

Contrary to Java, where there exists only primitive types and references, C++ is a language with value semantics. Assigning to a reference in Java only results in the actual memory modification of a memory address' size. The object that is no longer referred by the assigned-away reference is then left for garbage collection, if applicable. In C++, however, this means that assigning an object to another object of the same type results in the *assigned-to* object having the *assigned* object's state's copy within its own memory region. Traditionally, *copy assignment operators* have a "destructor" part where the current object's resources and buffers are released, and then a "copy constructor"-like logic where the copy of state takes place, however, the developer is free to choose a different implementation. The compiler-generated copy assignment operator implements a memberwise copy assignment for the entire object. Thus, the copy assignment operator is not generated by the compiler due to type infeasibility if one of the data members cannot be copy-assigned.

It is noteworthy to mention that not every language has defined the = assignment operator as an operator: in some languages, such as Ada or Pascal, assignment is defined as a statement/instruction, rather than an operator application. This has led to the inability to write copy assignment logic in Ada. To avoid use of assignment on types that are not designed for memberwise copy the `limited` keyword [18] and type-annotation is used.

In C++ it is commonly referred as *The Rule of Three* that if any of the copy constructor, copy assignment operator, and destructor is written explicitly by the developer, all of them should be written explicitly. This rule of thumb is not enforced by compilers but considered a good practice, because, as discussed earlier, explicitly specifying either will not stop the compiler from automatically creating the implicit definitions of the other special member functions.

2.4 Members for move semantics

The release of the C++11 Language Specification has introduced *move semantics*, which allows resources to be directly "stolen" by a variable from another, as opposed to a copy-constructed and the original data's memory destroyed. [13] This is used heavily with temporary objects which would get destroyed in the next statement. The move special members' default implementation executes a *move construction* or *move assignment* of every data member, however, the rules for their existence are more exquisite. Move members are not generated automatically if any explicit destructor, copy or move member exists, and an explicitly defined move member also turns off the automatic generation of copy members.

Accordingly, the Rule of Three has been extended to also include the two move members, and is referred to as *The Rule of Five*.

3 IMPLEMENTATION

3.1 Syntax transliteration

We used the open source LLVM/Clang Compiler Infrastructure for parsing and generation of special member visualisations because Clang's object-oriented *Abstract Syntax Tree* (AST) API allows for an optimised and maintainable application. An example subtree of the AST corresponding to the source code in Listing 1 can be seen in Listing 2. The *copy constructor*'s body corresponds to copying the right-hand record's single data member into the current record's corresponding data member.

```
CXXConstructorDecl </tmp/main.cpp:3:8>
|   implicit used constexpr A void
|   (const struct A &) noexcept inline
|-ParmVarDecl 20f90c0 used const struct A &
|-CXXCtorInitializer Field x int
| `-ImplicitCastExpr int <LValueToRValue>
|   `-MemberExpr const int lvalue .x
|     `-DeclRefExpr const struct A
|         lvalue ParmVar 20f90c0 ''
|         const struct A &
`-CompoundStmt

CXXConstructExpr <col:7, col:11> struct A
    void (const struct A &) noexcept
```

Listing 2: The Clang *AST* representation of the implicit copy constructor's body, and the call to it in `main()`.

Other compilers, might use different internal representations, on which these transformations would be infeasible to execute – in case of GNU/GCC, the *Register Transfer Language* (RTL) is only meant to be used by compiler-internal applications and code generation is organised into various steps called *loops*. The example of the same copy construction can be seen in Listing 3, which has already been stripped of semantic information and only the memory access for the data member can be studied from it by humans. It should be noted that the presented representation is the earliest and shortest where copy construction is apparent on the inner data member level. Previous transformation loops only show the copy constructor's call source line in it's original form, i.e. `A a2(a1);`.

```
(insn 7 6 8 2
  (set (mem/c:SI (plus:DI (reg/f:DI 82
      virtual-stack-vars)
    (const_int -8 [0xfffffffffffffff8]))
      [1 a2+0 S4 A64])
  (reg:SI 91)) "/tmp/main.cpp":6 -1
(nil))
```

Listing 3: The GNU *RTL* of the copy constructor call in line 9 of Listing 1.

We have utilised Clang's architecture to perform a parsing on the translation unit, and then performed a traversal on the built AST searching for all records, or a particular record with a name specified by the user. Once the record is found, we visit every special member's body, and in the case of constructors their *initialiser lists* [5] too. The AST nodes found in the subtrees of these nodes are then manually converted into a textual, source code representation.

```
struct A {
  A() {} // The default constructor.
  // The copy constructor.
  A(const A & rhs) : x(rhs.x) {}
};
```

Listing 4: The special members of the example class in Listing 1 translated back to source text.

There are three interesting cases that need to be noted, where explicit source code differs from what a compiler generates for itself automatically. First of all, the compiler generates the implicit members' arguments without an argument name. One such example can be seen in Listing 2, where the `ParmVarDecl` (parameter variable declaration) has no name, and the initialiser's `DeclRefExpr` (declaration reference expression) only refers to this `ParmVarDecl` by its memory address, `20f90c0`. Such a construct cannot exist in actual source code. As a remedy, we manually assign the name `rhs` to the variable – or in case multiple parameters would be possible, number them as `arg_1`, `arg_2`, ... – and use it in the pretty-printed code.

Another such interesting case is about move constructors and move assignment operators, namely that the compiler generates the argument as a temporary, an *xvalue*, from which move operations can be done. However, `T&& rhs` written in source code specifies a named variable, an *lvalue*, from whose members move must explicitly be specified by using a type annotation `std::move`, which casts the members to be *xvalue*s which denote variables that are essentially transformed into a temporary and their resources can be *moved from*. The pretty-printer annotates the right-hand sides of move initialiser or assignment expressions with `std::move` to ensure the same semantics. We only do this for record types, as no fundamental type supports move operations.

The third case is with regards to inheritance. In case a class has at least one superclass, the special members' default behaviour is to cast the current instance to the base class and call the appropriate constructor or assignment operator for each base class. A core principle in object-oriented programming is that *up-casting* – cast to any base class – is always possible and well-defined, however, this would result in unintelligible source code lines, such as `*this = rhs;` – which would lead to an infinite recursion if written in source code verbatim. The type system allows us to see that this = is for the base class, so we explicitly wrap the statement into a cast at the appropriate location to show base class initialisation to the developer. Examples of these cases are depicted in Figure 1.

We have encountered that the Standard only specifies generating a body for a special member if the currently compiled translation unit *ODR-uses* [7] the function. While no compiler error is given at compilation for an infeasible, *implicit deleted* special member unless used, the type system in Clang annotates the forward declaration of the function if it is deleted. Thus by using this annotation and the related diagnostics, we can, for each member without a body achieve either an explicit body generation or printing the reason behind the member being deleted by the type system in a single pass. It should be noted that generating the body for members which are

```
class Multiple
```

```
class Multiple : public D1, Base2
{

public:

Multiple()
  : D1()
  , Base2()
{
}

Multiple(const Multiple& rhs)
  : D1(static_cast<D1>(rhs))
  , Base2(static_cast<Base2>(rhs))
  , fooBar(rhs.fooBar)
{
}

Multiple(Multiple&& rhs)
  : D1(std::move(static_cast<D1>(rhs)))
  , Base2(std::move(static_cast<Base2>(rhs)))
  , fooBar(rhs.fooBar)
{
}

Multiple& operator=(const Multiple& rhs)
{
  static_cast<D1>(*this) = static_cast<D1>(rhs);
  static_cast<Base2>(*this) = static_cast<Base2>(rhs);
  fooBar = rhs.fooBar;
  return *this;
}

Multiple& operator=(Multiple&& rhs)
{
  static_cast<D1>(*this) = std::move(static_cast<D1>(rhs));
  static_cast<Base2>(*this) = std::move(static_cast<Base2>(rhs));
  fooBar = rhs.fooBar;
  return *this;
}

~Multiple();


};
```

Figure 1: Special member overview for a class with two base classes and a single `char` data member.

allowed to have one, and it is only an optimisation that generation didn't take place is a non-functional change and does not affect the semantics of the generated code – thus this transformation can safely be integrated into other compilation steps.

3.2 Special member overview

To facilitate better code comprehension, we have decided not only to show the implicit special members but every related overload of constructors and assignment operators. This allowed us to show a subset of the class' members which are related to the instance's lifetime.

The full overview proves useful when a special member is *defaulted*. If for example, a class contains some constructors and a user-defined copy constructor, the move members will not be generated automatically, however, the developer can explicitly ask the compiler to generate the methods with the implicit body rules by using the `= default` specifier, available in *C++11* and onwards. This is the suggested approach for modern C++, practised by most open-source projects. In this case, we show these members' body along with the rest of the class with the annotation that the user requested the body generation.

Another case for the full view is showing the reason why a special member was not automatically generated by printing a hint from the semantic analysis' diagnostics.

4 CONCLUSION

In this paper, we have discussed the rules and behaviour of automatically generated special member functions, an intrinsic feature of the C++ programming language. We have introduced an approach to transliterate the compiler's internal representation of these special members to source text to promote understanding of software projects without resorting to unfavourable techniques such as reading syntax trees manually.

We have implemented our solution in the open-source code comprehension tool CodeCompass [1, 14, 15] — `http://github.com/Ericsson/CodeCompass` — as an additional visualisation over C++ files. The upstreaming of this addition is underway at the writing of this paper.

ACKNOWLEDGMENTS

This work presented in this paper was supported by the European Union, co-financed by the European Social Fund in project `EFOP-3.6.3-VEKOP-16-2017-00002`.

REFERENCES

[1] CodeCompass. 2012. A software comprehension tool for large-scale software written in C/C++ and Java. http://github.com/Ericsson/CodeCompass
[2] Margaret Ellis. 1990. *The Annotated C++ Reference Manual.* Addison-Wesley, Reading, Massachusetts, USA.
[3] James Gosling, Bill Joy, Guy L. Steele, Gilad Bracha, Alex Buckley, and Daniel Smith. 2017. *Finalization of Class Instances* (1st ed.), Chapter 12.6, 389–393. In [4]. https://docs.oracle.com/javase/specs/jls/se9/jls9.pdf visited on 2018-08-13.
[4] James Gosling, Bill Joy, Guy L. Steele, Gilad Bracha, Alex Buckley, and Daniel Smith. 2017. The Java Language Specification, Java SE 9 Edition. https://docs.oracle.com/javase/specs/jls/se9/jls9.pdf visited on 2018-08-13.
[5] ISO. 2012. *Initializing bases and members*, Chapter 12.6.2, [class.base.init]. In [6]. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50372
[6] ISO. 2012. *ISO/IEC 14882:2011 Information technology — Programming languages — C++, version 11 (C++11).* International Organization for Standardization, Geneva, Switzerland. 1338 (est.) pages. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50372
[7] ISO. 2012. *One definition rule*, Chapter 3.2.3, [basic.def.odr]. In [6]. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50372
[8] ISO. 2012. *Special member functions*, Chapter 12, [special]. In [6]. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50372
[9] ISO. 2012. *Temporary objects*, Chapter 12.2, [class.temporary]. In [6]. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50372
[10] Marián Juhás, Zoltán Juhász, Ladislav Samuelis, and Csaba Szabó. 2009. Measuring the complexity of students' assignments. *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae.* 31 (2009), 203–215.
[11] Zoltán Juhász, Marián Juhás, Ladislav Samuelis, and Csaba Szabó. 2008. Teaching Java programming using case studies. *Teaching Mathematics and Computer Science.* 6(2) (2008), 245–256.
[12] Stanley Lippman. 1996. *Inside the C++ Object Model.* Addison Wesley Longman, Reading, Massachusetts, USA.
[13] Scott Meyers. 2015. *Effective Modern C++: 42 specific ways to improve your use of C++11 and C++14.* O'Reilly Media, Sebastopol, California, USA.
[14] Zoltán Porkoláb and Tibor Brunner. 2018. The CodeCompass Comprehension Framework. In *Proceedings of the 26th Conference on Program Comprehension (ICPC '18).* ACM, New York, New York, USA, 393–396. https://doi.org/10.1145/3196321.3196352
[15] Zoltán Porkoláb, Tibor Brunner, Dániel Krupp, and Márton Csordás. 2018. CodeCompass: An Open Software Comprehension Framework for Industrial Usage. In *Proceedings of the 26th Conference on Program Comprehension (ICPC '18).* ACM, New York, New York, USA, 361–369. https://doi.org/10.1145/3196321.3197546
[16] The LLVM Project. 2003. *Clang*: C Language Family Frontend for LLVM. http://clang.llvm.org visited on 2018-08-13.
[17] Bjarne Stroustrup. 1994. *The design and evolution of C++.* Addison-Wesley, Reading, Massachusetts, USA.
[18] S. Tucker Taft, Robert A. Duff, Randall L. Brukardt, and Erhard Ploedereder. 2000. *Consolidated Ada Reference Manual: Language and Standard Libraries.* Springer-Verlag, Berlin, Heidelberg, Germany.

How Does an Integration with VCS Affect SSQSA?

Bojan Popović
Naovis d.o.o.
Bulevar oslobođenja 30A
Novi Sad, Serbia
bojan.popovic@primafin.com

Gordana Rakić
University of Novi Sad, Faculty of Sciences Trg
Dositeja Obradovića 4
Novi Sad, Serbia
goca@dmi.uns.ac.rs

ABSTRACT

Contemporary trends in software development almost necessarily involve version control system (VCS) for storing and manipulation of source code and other artifacts. Consequently, tools supporting the development process such are software analysis tools integrate with VCS. In most of cases tools support only analysis of the resources in VCS repositories, while some of them rely on VCS to improve analysis process and results. In this paper we explore how an integration of the SSQSA platform with VCS influences some of its performances.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

Software quality analysis, intermediate representation, Version Control System

1. INTRODUCTION

Quality of a software product is observed through the level of satisfied requirements. It could be assessed by its execution by applying different techniques of dynamic analysis. These techniques are applicable when the product is ready for testing which might be late to recognize weaknesses or issues. On the other side, static analysis techniques are traversing source code and its various intermediate representations which makes them applicable already in the early phases of software development process [5].

Contemporary software development practice relies on source code repositories and their synchronization implemented by various version control systems (VCS). VCS are used to store the whole history of activities in the evolution of a software product, from version information to the finest details about every individual change in the repository, including information about contributors to the changes.

Consequently, software analysis tools integrate support for VCS. Usually this support means possibility to analyze code stored to VCS repositories. In some cases tools also rely on advantages of VCS to improve analysis performances or results.

In this paper we explore potential advantages of integration of SSQSA (Set of Software Quality Static Analyzers) platform [9] with GIT [2] as a representative VCS. First, we introduce a concise background by describing VCS (Section 2) and SSQSA (Section 3). Prerequisites for the integration and the integration are described in the section 4. We discuss results in the section 5 and possible application models and scenarios in the Section 6, that is followed by comparison to related integration solutions (Section 7). We conclude the paper in the Section 8. This paper is summary of a master thesis described in [8] (in Serbian).

2. VERSION CONTROL SYSTEMS

Version control systems (VCS) might have very broad application in different areas of content manipulation for personal or professional purposes. These are tools used primarily to support teams and individuals in development and maintenance of a software products. These systems remember all the changes of separate files, so that at any time we can recover a specific version, or follow and compare changes over the time. In this way, all data is safer, good synchronization between the team members is ensured, the possibilities for errors are significantly reduced, and therefore the project development process is improved.

VCS are divided into two large groups [2]:

CVCS: Centralized Version Control System where all the data are stored to a centralized server. This approach is certainly easier to maintain, but in case of system failure, all information about the project will be lost. Additionally, availability of a network connection is very important. Previously, this was the standard way to execute version control. Representatives of this group are CVS: Concurrent Versions System [4] and Subversion [3].

DVCS: Distributed Version Control System where clients map the whole repository. If a server failure occurs, any of the client repositories can be copied back to the server to restore it. However, local copy enables us to work on changes independently of a network connection while

Property	Git	Mercurial
Simple GUI	-	+
Getting started for beginners	-	+
Simplicity branches visualization	-	+
Speed (Windows OS)	-	+
Speed online	+	-
Changing the history	+	+
Using the index	+	-
PL independent extensions	+	-
Repo. migrating to another system	+	-

Table 1: Comparison between Git and Mercurial

the connection is necessary only for saving changes at the remote repository or taking a version from it. Files stored on the hard disk are of small size, and hence this does not pose a problem problem of a storage space.

An additional advantage of DVCS is that we can share the changes with other team members before they are shared globally. On the other hand, there is little advantage of centralized systems compared to distributed ones. Centralized systems offer us an easier way to control all the people who access the server, as well as easy provision of a central point where all the changes are in place. They also offer us the option of downloading only a piece of code, if we only need to work on a project module. However, if needed, one copy of the project in the DVCS can be announces as the main one, and thus we can simulate the centralized system.

Distractions that can be addressed to distributed systems are more technical. For example, in case of a project with many large files that can not be compressed, more storage space is required. Additionally if we are working on a large project that contains many customized changes, downloading a full version of the project can take longer than expected, and also take up more space on the hard drive than expected.

All described differences bring to the decision to conduct the first experimental integration SSQSA platform with a DVCS. Therefore, we compare Git [2] and Mercurial [6] as the main representatives of DVCS in order to compare their properties to our requirements (Table 1). We can conclude that Mercurial has better characteristics from the users point of view, but for our integration these characteristics do not have value. On the other hand easiness to integrate with other systems, possibility to migrate to an other system and speed are extremely important to us. Therefore, in this work, we integrate SSQSA with Git.

3. THE SSQSA PLATFORM

The SSQSA (Set of Software Quality Static Analyzers) [9] is a set of tools that enables language independent static software product analysis based on its source code. Language independence is ensured by a universal intermediate representation of a source code called eCST (enriched Concrete Syntax Tree). Once when this representation is produced for any system, written in any set of programming languages, it can be transformed to some of derived intermediate representations such are dependency networks, at different abstraction levels, or flow graphs. The fact that derived representations are generated based on eCST, by a unique implementation of the derivation process, ensures their language independence and universality, too.

By traversing all or some of these universal intermediate representations different analysis algorithms are implemented. Therefore, it is possible to have a single implementation of every functionality that we integrate in the SSQSA which ensures consistency of the results across different languages, but also adaptability to a new language and extendability by a new analysis [9]. Described process and a corresponding platform design is illustrated by the Figure 1.

Current version of the SSQSA platform manipulates input source code from an local directory (components colored by gray color), while our primary goal in this research is to integrate it to analyze the code stored in a Git repository. Additionally, we will explore how usage of Git repository for storing intermediate representation affects SSQSA platform and its performances. This level of the integration will enable us to traverse only changed fragments of the structures, which might further lead to improvement of performances of the analyses. The first prototype includes only results of generation of eCST in the repository. New components that implement integration are yellow-colored in the Figure 1.

4. THE SSQSA AND GIT INTEGRATION

To enable collaboration of SSQSA with Git, it was necessary to connect eCSTGenerator to Git repository and to enable it to process the source code stored in it. After the first connection eCSTGenerator is processing the whole content of the repository and generates its eCST representation. Every next time, eCSTGenerator will process only changed files. This feature was not easily implementable before the integration with Git.

In addition, SSQSA uses advantages of its integration with Git at one more level. Namely, after the set of eCST is generated, it is stored to a Git repository so that other components can also process only changes between versions. For these purposes we do not use the same repository as it is a dedicated development repository, while developers do not have to be affected by the analysis.

5. RESULTS

To explore applicability aspects of the described integration solution, we measure time needed for generation of eCST representation of a JavaScript project "proton-native"[1].

First, we observe time needed only for generation of eCST representation of the source code from the local folder and compare it to the time needed to generate it for the code stored in a Git repository (Table 2).

As we can see, for the first commit generation process lasted for significantly longer time. The reason for this is time needed for the connection to the Git repository. However, even though process spends additional time on the connection, in later commits we get better results from the version integrated with Git.

[1]https://github.com/kusti8/proton-native

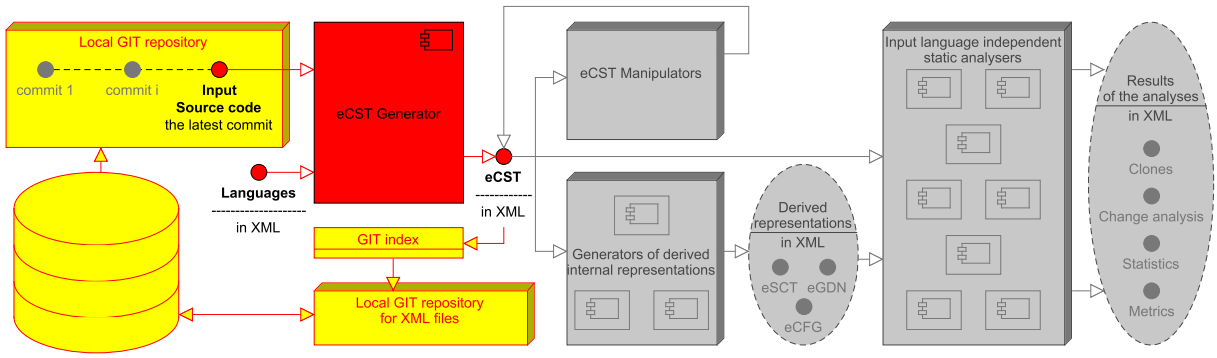


Figure 1: SSQSA platform and its integration with Git

Version no. of commit	from a local dir	from a Git repo	Time for the Git connection
7.	744 ms	1250 ms	720 ms
14.	812 ms	1270 ms	754 ms
34.	1589 ms	1353 ms	739 ms
80.	1601 ms	1520 ms	870 ms
126.	1650 ms	1515 ms	780 ms

Table 2: Comparison between time needed for eCST generation proccess from a local directory and from a Git repository.

Eventually, if we include functionality for committing of generated eCST to a repository, time needed for whole process goes over 6000ms. Obviously, in this scenario integration reduces performances of SSQSA. Still, further integration will utilize benefits of version control to improve generation of derived intermediate representations. Finally, it will be integrated with the analyzers. It can be expected that, with the growth of data that will be saved up in the exchange, traverse and analysis process, the benefits from the integration will also grow. Therefore, effects of the integration on other components still have to be explored (Section 8.

6. APPLICATION SCENARIOS

Depending on a scenario, Git has three common application models: a *centralized model*, a *pull-request model*, and *a Director and Lieutenants model*[2]. In a centralized system, all members of the team synchronize their changes in a central repository that stores all source code. In the pull-request model, developers can make changes to his local repository, and he commits them to his own repository, and can see the changes that other team members make. In this model one repository is considered the main repository. In order to accomplish the changes in it, a request is sent to the project leader to pull the changes. The project leader can add developer's repository as a remote repository, locally test changes, and if everything is fine, save them to the main repository. In a Director and Lieutenants model the project is divided into sub-projects and distributed among teams. Each team (sub-project) has its own repository and its leader, and usually works according to the pull-request model on the local level.

The most practical model for implementing a new implementation for the use of Git is the pull-request model. A project leader can start an eCST generator on a new repository commit to analyze the modified file. If a developer wants to create XML trees, it can also launch an eCST generator at each commit. The problem can arise if more teams are made and the eCST generation process is lunched then only. In this case it must be adapted to go through all the commits, not only looking at the latest changes.

The "Director and Lieutenant" model is also suitable for new implementation. Each sub-project has its own leader who can create XML trees. Also, the leader of the repository may generate eCST when joining new changes to a branch of a project (merge). Also, if developers want to generate XML trees, the same rules apply as with the Pull-Request model.

The centralized model is the most unpractical model for using the new implementation. All team members commit their changes to a centralized repository, which in this case contains a lot of commits through which traversal should be conducted.

7. RELATED SOLUTIONS

Many tools also support code analysis from various VCS such as BCH: Better Code Hub[2] and SonarQube[3], primarily because the repositories have become the standard code storage. However, only some tools rely on versions for more advanced analysis.

Lean Language Independent software analyzer (Lisa) is a software that analyzes the quality of software projects. The main goal of Lisa is to analyze a large number of project revisions asynchronously with minimal redundancy. Analyses are aimed to cover as many analyzes, and as many program-

[2]https://bettercodehub.com/
[3]https://www.sonarqube.org/

ming languages as possible. These goals are comparable with the goals of SSQSA, as well as the new implementation presented in this paper. However, Lisa currently supports three programming languages, while the SSQSA framework currently allows us to work with more than ten programming languages. Concerning the subject of this paper, We can note certain differences in the approach to the problem and the concrete solution implementation. For the needs of the Lisa analyzer, a special interface called SourceAgent has been developed. It supports the asynchronous access to the Git repository and file revisions [1]. On the other hand, SSQSA, with the current implementation, uses all the benefits of the Git and the library for interactions with it, looks at the differences between the last two committees, and reads all the files that have been changed, and generates XML trees for them. Furthermore, Lisa communicates directly with the Git repository by making a local copy of the remote repository to a local hard disk, while our implementation allows reading from a local disk and thus does not require an internet connection. Internet connection is only needed if we want to save the generated XML tree in a remote repository.

The Analizo is a solution that analyzes source code written in different programming languages, whose emphasis was on C, C ++ and Java. The analysis supports the reading of content from remote repositories for each audit in which the source code has been changed in the project [10] and, unlike the SSQSA which currently allows reading of contents only from the Git repository, allows reading from the Git and Subversion repositories, and then generates CSV files. SSQSA also compares file revisions and decides from which files to create an XML tree. An advantage over Analizo is that we can monitor file versions on a remote repository. Again, the difference is in the number of supported languages: Analyzo supports three languages, while SSQSA currently supports more than ten programming languages.

EvoJava is a tool for static code analysis of an input from a Java repository. It uses a VCS to access the code, mines the source repository, and calculates metrics. Unlike the SSQSA platform, EvoJava uses Subversion (SVN) and processes only *.java* files. The output file is also in .XML format, but containing metric results. EvoJava takes a list of the code versions that is in the repository and thus creates a model based on the XML-generated files [7]. SSQSA, on the other hand, observes the latest changes that are committed to a remote repository, finds these files in the file system and creates XML files based on them. Later it automatically commits them to a special local or remote repository, where we can track what changes were made during the evolution of our software. We cane also note the variety in supported programming languages in SSQSA while EvoJava only supports Java programming language.

8. CONCLUSION AND FUTURE WORK

Following actual trends in software development and software analysis SSQSA frameworks goes into a direction of integration with VCS. In this paper we compare characteristics of different VCS and select Git as a first candidate for the integration. Furtehr, we describe its integration with Git and explore possible benefits from this integration for the performances of the platform.

Integration is developed at two levels. At the first level the platform is connected to the Git repository in order to enable processing source code stored in it. At the next level of the integration we use Git repository to store XML file containing eCST intermediate representation of the source code so that we can always look only for changes, and not traverse all the code, or more precisely, eCST representation of it. This is very important if we have in mind that one input file (compilation unit) is represented by one eCST.

At the first look, the results of the integration are not promising. Namely, Git connection used the time that we can save by looking only in the changes and not in the whole source code. However, Without storing trees to the Git repository we are already saving some processing time. In case when we store eCST in a Git repository we are spending more time but in the future work we will explore if this cost may be payed off after extending this integration on generation of derived representations and analyzers. For example, generation of dependency network currently traverses all the trees while after the full integration with Git it will also look only for changes. The similar expectation we have from an integration of analyzers with Git. Therefore, these integration activities will be subject of the a future work, as well as analysis of potential costs and benefits, and selection of the most suitable usage scenarios.

9. REFERENCES

[1] C. V. Alexandru, S. Panichella, and H. C. Gall. Reducing redundancies in multi-revision code analysis. In *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*, pages 148–159. IEEE, 2017.

[2] S. Chacon and B. Straub. *Pro git*. Apress, 2014.

[3] B. Collins-Sussman, B. W. Fitzpatrick, and C. M. Pilato. Version control with subversion, 2006. *Accessible in URL: http://svnbook. redbean. com*, 2007.

[4] D. Grune et al. *Concurrent versions systems, a method for independent cooperation*. VU Amsterdam. Subfaculteit Wiskunde en Informatica, 1986.

[5] G. O Regan. *Introduction to software quality*. Springer, 2014.

[6] B. O Sullivan. *Mercurial: The Definitive Guide: The Definitive Guide*. "O'Reilly Media, Inc.", 2009.

[7] J. Oosterman, W. Irwin, and N. Churcher. Evojava: A tool for measuring evolving software. In *Proceedings of the Thirty-Fourth Australasian Computer Science Conference-Volume 113*, pages 117–126. Australian Computer Society, Inc., 2011.

[8] B. Popović. Integration of a platform for static analysis with a version control system (in serbian). Master's thesis, Faculty of Sciences, University of Novi Sad, 2018.

[9] G. Rakić. *Extendable and adaptable framework for input language independent static analysis*. PhD thesis, Faculty of Sciences, University of Novi Sad, 2015.

[10] A. Terceiro, J. Costa, J. Miranda, P. Meirelles, L. R. Rios, L. Almeida, C. Chavez, and F. Kon. Analizo: an extensible multi-language source code analysis and visualization toolkit. In *Brazilian conference on software: theory and practice (Tools Session)*, 2010.

Indeks avtorjev / Author index

Konferenca / Conference

Uredil / Edited by

Sodelovanje, programska oprema in storitve v informacijski družbi / Collaboration, Software and Services in Information Society

Marjan Heričko