Consortium for Software Engineering Research
2019 Fall Meeting

November 3rd, 2019 - Markham, ON

LIST OF POSTERS

Name	Affiliation	Submission Title	Abstract	Co-Authors (comma separated)
Mohammadreza Rasolroveicy	Polytechnique Montreal	Balancing performance, security and cost tradeoffs of Blockchain in IoT applications: A comparative study.	Internet-of-Things (IoT) technologies have gained prevalence in a number of domains including Smart Vehicles, Smart Buildings, Smart Health and others. IoT applications in such domains put emphasis on security, privacy and trust, which can be characterized as sensitive issues. Blockchain technologies, which acts as an immutable and transparent public record of data secured using a P2P (peer-to-peer) network, have emerged as potential solutions to address these issues. However, Blockchain can be inefficient both in terms of time and cost due to high bandwidth overhead and delays. In this paper, we will study three different popular Blockchain platforms to see if there is a single best with respect to time and computation overheads , or if there are tradeoffs between the three platforms for IoT applications.	Marios Fokaefs
Mahsa Hadian	Phd student at polytechnique of montreal	Internet-of-Things solutions for Smart Campuses: A comparative study of clustering techniques	Today, Internet-of-Things solutions for Smart Buildings, like instrumentation of buildings with sensors and digitally controlled equipment, are on the rise. Recent advancements in relevant technologies have enabled such applications. Their benefits have become immediately apparent with reductions on cost and energy consumption, and the increase of comfort and productivity for residents and personnel of such buildings. The proposed research aims to study software solutions for the management of Smart Buildings with the goal to optimize energy consumption and costs associated with both the sensing and controlling equipment as well as the computation infrastructure, with respect to the performance of data analytics. More specifically, our research focuses on managing Smart Campuses like universities and hospitals, which increases the complexity of the problem. The research question is how to coordinate these different buildings to optimize energy consumption and increase the capabilities of entire campuses. For this purpose, we employ adaptive clustering to distribute and organize data analytics for the campus and we exploit concepts from Edge computing and Hierarchical clouds. In this work, we present a comparative study of different clustering algorithms to find optimal one across different quality indices concerning the management of Smart Buildings.	
Felipe Rivera	University of Victoria	Towards Continuous Monitoring in Personalized Healthcare through Digital Twins	Risks associated with the effects of persistent chronic diseases demand continuous and effective monitoring mechanisms. Moreover, particular patient conditions require medical treatments and care to be tailored according to individual needs. However, these considerations impose new challenges on the engineering of systems that support these required capabilities. Research on the application of the novel concept of Digital Twin (DT) in healthcare might provide the means to overcome these challenges. DTs might represent a significant step forward towards the achievement of effective continuous monitoring of chronic diseases and personalized healthcare. This proof of concept presents our vision and emerging results in the application of DT practices in the healthcare domain. For this, we analyze the relevance of using DTs in the disease management of diabetes.	Norha M. Villegas, Hausi A. Müller
Batyr Nuryyev	University of Alberta	Re-engineering Eclipse OMR's variability implementation mechanism	Eclipse OMR is an open-source framework for building language runtimes. It consists of C++ components such as compiler, garbage collection, threads, etc. that could be used to build a language runtime. OMR uses static polymorphism and extensible classes as its variability mechanism due to its high runtime performance requirement. However, there are several major design disadvantages that impede the project's simplicity, usability and extensibility; for example, developers who want to consume OMR find it difficult to figure out which classes should be extended, and which are used only by OMR itself. Our work focuses on the challenges posed by the current variability implementation mechanism. Our goal is to explore existing software variability mechanisms for similar (and/or related) projects as well as their potential to be successfully applied to Eclipse OMR.	
Mahmoud Alfadel	Concordia University	On the Unexploitability of Security Vulnerabilities: A Case Study on Node.js	Software vulnerabilities have a large negative impact on the software systems that we depend on daily. Reports on software vulnerabilities always paint a grim picture, with some reports showing that 83% of organizations depend on vulnerable software. However, our experience leads us to believe that, in the grand scheme of things, these software vulnerabilities may have less impact than what is reported. Therefore, we perform a study to better understand the exploitability of software vulnerabilities. We define three levels of exploitability for vulnerabilities, based on their lifetime. Then, we perform an empirical study involving 11,288 real-world, active, and mature open source Node.js applications. Our findings show that although 51.1% of the examined applications depend on at least one vulnerable package, 87.1% of these vulnerabilities have a low chance of being exploited. Moreover, we find that in the case of highly exploitable vulnerabilities, it is the application's lack of updating that makes them vulnerable, i.e., it is not the existence of the vulnerability that is the real problem. Finally, we verify our findings at different stages of the application's lifetime and find that our findings still hold. Our study argues that when it comes to software vulnerabilities, things may not be as bad as they seem and that considering exploitability of a vulnerability is key.	

Consortium for Software Engineering Research
2019 Fall Meeting

November 3rd, 2019 - Markham, ON

Name	Affiliation	Submission Title	Abstract	Co-Authors (comma separated)
Ahmad Abdellatif	Concordia University	A Comparison of Natural Language Understanding (NLUs) Engines for Software Bots	Organizations are investing heavily in software bots to assist developers with their daily tasks. At the heart of every bot is an NLU component, that enables the bot to understand natural language input. Developers of bots have a choice when it comes to NLUs, e.g., Rasa, Dialogflow, etc. However, which NLU is best to use in the context of SE remains an open question. Therefore, we evaluate the most popular NLU services in terms of intent classification, NLUs' confidence score, and entity extraction using two datasets to compare their performances. Our findings show that IBM Watson NLU provides superior performance amongst its competitors. Also, our findings highlight areas that researchers should focus on to improve their NLUs.	Khaled Badran, Emad Shihab
Max Ellis	University of Alberta	Re-visiting Refactoring-aware Software Merging	Dealing with merge conflicts is one of the main challenges developers face while collaborating on modern version control systems. Resolving merge conflicts is a time-consuming process which distracts developers from their main tasks. The current software merging tools, integrated in modern version control systems, are typically text-based and do not understand the underlying code changes. As previous work shows, refactorings (software changes that may alter the syntax of the program but not its semantics) typically cause more complicated merge conflicts in git, even though they have the potential to be automatically resolved. Dig et al. previously proposed a refactoring-aware merge conflict resolution algorithm for Java; however, the approach was never integrated into any modern version control systems or evaluated at scale. In this work, our goal is to re-implement and adapt this algorithm for git and to evaluate it on a large scale.	Sarah Nadi
Sunil Kumar	University of Victoria	Virtual Machines migration detection on cloud infrastructures	Movement of Virtual Machines (VMs) from one cloud resource to another is called VM migration. There could be various reasons to do VM migration including service load balancing and host maintenance and failure. Most of today's advanced cloud infrastructure providers like OpenStack and VMware support VM migration operations, including Cold migration i.e., migration of a powered-off VM and Live migration i.e., migration of powered-on VM. Nowadays systems are deployed to multiple clouds to avoid a single point of failure. Therefore, inter-cloud migrations are becoming more relevant. Despite cloud providers offering migration APIs, there is no way to detect current inter-cloud migrations or even list previous ones. Our work investigates ways to detect inter-cloud, as well as intra-cloud VM migrations including the detection of the type of migration. The details of migration also will include which part of the VM was migrated i.e. Compute (CPU), Disk or Network.	Miguel Jiménez, Hausi Müller.
Miguel Jimenez	University of Victoria	A Model-based Middleware for Continuous Software Evolution at Run-time	Infrastructure-as-Code (IaC) enables provisioning and managing dynamic infrastructure resources through machine-readable configuration files. It is also referred to as programmable infrastructure in reference to the adaptation and application of practices and tools from software engineering on IT infrastructure management. Changes to the computing infrastructure and/or the execution environment are made in a structured way, through reliable and established processes. Therefore, tools that are alien to such processes hinder the repeatability of the provisioning and management. This posits a challenge for organizations maintaining legacy systems that have already developed hundreds of scripts and tools. The lack of repeatability in their deployment practices makes it difficult to adopt tools such as IBM Cloud Automation Manager, which relies on the use of Terraform--an IaC tool. In this exhibit, we introduce a middleware for the continuous evolution of IaC specifications. Our middleware automatically monitors changes on cloud resources and updates structured specifications in a code repository. To do so, we rely on practices of continuous software engineering. Therefore, all changes committed to the repository follow the usual development workflow.	Miguel Jiménez, Hausi Müller, Gabriel Tamura, Joe Wigglesworth, Ian Watts
Prashanti Priya Angara	University of Victoria	Quantum Problem Solving and Algorithm Design on the IBM Q Platform	As a part of our IBM CAS project, we focus on quantum computing and quantum algorithms. We concentrate on three avenues: (1) develop a toolkit to identify Q problems and Q hybrid problems; (2) develop Q problem solving and algorithm design techniques for selected application domains; and (3) develop Q software for selected problems using the IBM Qiskit open-source quantum computing framework. In this poster/demo, we show our first steps into the exciting realm of quantum computing by exhibiting fundamental concepts in quantum computing, including superdense coding, teleportation and quantum Fourier transform, and their implementations in Qiskit Terra and Aqua.	Hausi Muller, Ulrike Stege
Scott Curtis Brisson	University of Toronto	Understanding Collaboration in Software Repositories	GitHub facilitates software development practices that encourage collaboration and communication. Much of this communication occurs between a software repository, its forks, and its sub-forks. In this poster, we present our findings from an analysis of communication behaviors within the context of 385 repository groups, and relate them to project success. These communication behaviors include, but are not limited to, pull requests, issues and "following" behaviors, as well as a sentiment analysis of comments, and a topic analysis on issue discussions. These results provide insight on the importance of communication that takes place between software repositories.	Ehsan Noei, Kelly Lyons

Consortium for Software Engineering Research
2019 Fall Meeting

November 3rd, 2019 - Markham, ON

Name	Affiliation	Submission Title	Abstract	Co-Authors (comma separated)
Enning Zhang	University of Toronto	Designing for knowledge work in the age of AI: An industry study	To evaluate publicly listed companies for their environmental, social and governance performance, sustainability analysts rely on semi-automated data-intensive workflows that acquire and analyze complex, heterogeneous data sources using Machine Learning algorithms. They interpret the algorithmic results to judge their relevance and use their substantive content in a new kind of data-intensive knowledge work that fuses AI with individuals' professional and curatorial expertise and judgment. The analysts' expertise and judgment are as irreplaceable as the analytic capabilities provided by Machine Learning. How does this new phenomenon affect systems design? This poster describes an ongoing collaboration between a company R&D team and an academic research team to examine how analysts can be better supported in judging the relevance of the outcomes of classification algorithms produced by curation mechanisms powered by Machine Learning. We apply Soft Systems Methodology to develop a shared understanding of the problem situation across stakeholders, and we use the Research Object framework (http://researchobject.org) to model the information products that analysts create and their computational provenance. The project insights inform the iterative development of the companies' AI-powered data acquisition and curation platform.	Rachel Booth, Andres de los Rios, Christoph Becker, Fabian Fagerholm
Lizhi Liao	Concordia University	Using Black-Box Performance Models to Detect Performance Deviations under Variable Workloads from the Field: An Empirical Study	Performance deviations of large-scale software systems often lead to both financial and reputational losses. In order to detect performance deviations, performance tests are typically conducted in a testing (non-production) environment, using test suites with predefined workloads. Afterward, performance models are used to check whether a software version has a performance deviation against an earlier version. However, the real workloads in production are constantly changing, making it unrealistic to test all possible workloads in the field. More importantly, performance testing is usually very expensive as it requires extensive resources and lasts for an extended period. In this paper, we build performance models using the readily-available field data from the past to detect field performance deviations in the future. Practitioners can leverage such performance models to replace resource-heavy performance tests that may not even be realistic. Based on the historical field data, we build black-box performance models that capture the relationship between the performance of a software system (e.g., CPU usage) under real field workloads and its runtime activities that are recorded in the ready-available logs. Then, we use our models to predict the expected performance of the software system under new workloads and compare the expected performance with the measured performance to detect performance deviations. In this paper, we conducted a case study on two open source systems, i.e., OpenMRS and Apache James, and one large-scale industrial system. Our study results show that the black-box performance models can successfully detect performance deviations under unseen variable workloads from the field. Our approach has been adopted by the industrial system to detect performance deviations on a daily basis.	Lizhi Liao, Jinfu Chen, Heng Li, Yi Zeng, Weiyi Shang, Jianmei Guo, Catalin Sporea, Andrei Toma, Sarah Sajedi
Lama Moukahal	PhD Student at Queen's University, Canada	Security Vulnerability Metrics for Connected Vehicles	Software integration in modern vehicles is continuously expanding. This is due to the fact that vehicle manufacturers are always trying to enhance and add more innovative and competitive features that may rely on complex software functionalities. However, these features come at a cost. They amplify the security vulnerabilities in vehicles and lead to more security issues in today's automobiles. As a result, the need for identifying vulnerable components in a vehicle software system has become crucial. Security experts need to know which components of the vehicle software system can be exploited for attacks and should focus their testing and inspection efforts on it. Nevertheless, it is a challenging and costly task to identify these weak components in a vehicle's system. In this presentation, we propose some security vulnerability metrics for connected vehicles that aim to assist software testers during the development life-cycle in order to identify the frail links that put the vehicle at high security risks. Vulnerable function assessment can give software testers a good idea about which components in a connected vehicle need to be prioritized in order to mitigate the risk and hence secure the vehicle. The proposed metrics were applied to OpenPilot - a software that provides Autopilot feature - and has been integrated with 48 different vehicles. The application shows how the defined metrics can be effectively used to quantitatively measure the vulnerabilities of a vehicle software system.	Mohammad Zulkernine
Anika Anwar	Queen's University	Cloud-based Sybil Attack Detection Scheme for Connected Vehicles	As a part of an Intelligent Transportation System (ITS), connected vehicles provide useful information to drivers and the infrastructure to help make safer and more informed decisions. However, vehicle connectivity has made the ITS more vulnerable to security attacks that can endanger vehicle's security as well as driver's safety. Sybil attack is a very common attack, considered dangerous in a distributed network with no centralized authority. When launched against connected vehicles, it consists of controlling a set of vehicles with forged or fake identities to try to alter the measurements and data collected by the ITS, leading to sub-optimal decisions. In this paper, we provide a cloud-based detection scheme for connected vehicles against such an attack. Contrary to the previous distributed solutions in the literature, this paper presents a cloud-based solution that integrates a cloud-based authorization unit to authenticate legitimate nodes using symmetric cryptography and real-time location tracking.	Dr. Md. Zulkernine, Dr. Talal Halabi

Consortium for Software Engineering Research
2019 Fall Meeting

November 3rd, 2019 - Markham, ON

Name	Affiliation	Submission Title	Abstract	Co-Authors (comma separated)
Max Lamothe	Concordia University	Automatically Assisting Android API Migrations Using Code Examples	The fast-paced evolution of Android APIs has posed a challenging task for Android app developers. To leverage Androids frequently released APIs, developers must often spend considerable effort on API migrations. Prior research and Android official documentation typically provide enough information to guide developers in identifying the API calls that must be migrated and the corresponding API calls in an updated version of Android (what to migrate). However, API migration remains a challenging task since developers lack the knowledge of how to migrate the API calls. There exist code examples, such as Google Samples, that illustrate the usage of APIs. We posit that by analyzing the changes of API usage in code examples, we can learn API migration patterns to assist developers with API Migrations. In this poster, we propose an approach that learns API migration patterns from code examples, applies these patterns to the source code of Android apps for API migration, and presents the results to users as potential migration solutions. To evaluate our approach, we migrate API calls in open source Android apps by learning API migration patterns from code examples. We find that our approach can successfully learn API migration patterns and provide API migration assistance in 71 out of 80 cases. Our approach can either migrate API calls with little to no extra modifications needed or provide guidance to assist with the migrations. Moreover, through a user study, we find that adopting our approach can reduce the time spent on migrating APIs, on average, by 29\%.	Ian (Weiyi) Shang, Peter (Tse-Hsun) Chen
Hetong Dai	SENSE, Concordia University	Natural or Unnatural? An Automated Log Parsing Approach Based on Entropy	Logs are important resources that record the runtime information of software systems. Practitioners leverage log analysis to support various software operations and maintenance efforts, such as anomaly detection. Log parsing is often the first step of such log analysis method. The goal of log parsing is to distinguish the static template part and the dynamic part in a log line. As the source code that specifies the static templates of logs is usually inaccessible in practice, log parsing is a challenging task for practitioners and researchers. In this work, we propose an automated log parsing approach that leverages the entropy of each token in a log line to distinguish static and dynamic tokens. Out intuition is that the static part of a log line is as natural as the natural languages (i.e., with low entropy) as it is written by human beings, while the dynamic part is unnatural (i.e., with high entropy) as it is dynamically produced by machines. Specifically, we leverage n-gram language models to capture the entropy of each token in a log line. Then, we classify those tokens with low entropy as static tokens and those with high entropy as dynamic tokens. The evaluation of our approach demonstrates both high accuracy and efficiency that outperforms existing approaches.	Heng Li, Weiyi Shang
Yuyang Liu	University of Toronto	Do README files follow GitHub guidelines?	README files are usually the first item users read when visiting a project repository. GitHub provides guidelines for the structure and content of README files. For example, it is recommended that README files include the following sections: a project title, description, a table of contents to help navigate the README file, installation information, usage instructions (which could include screenshots), contribution instructions, author credits, and license information. In this poster, we present an analysis of 14,901 open source Java projects on GitHub to investigate how closely README files align with GitHub guidelines. We also report on how the inclusion of certain README sections recommended by GitHub relate to project popularity (as measured by the number of stars).	Ehsan Noei, Kelly Lyons