Consortium for Software Engineering Research
2019 Fall Meeting

November 3rd, 2019 - Markham, ON

PROGRAM

Time	Session	Chair	Speaker	Title	Abstract
8:30	Registration + Breakfast				
9:00	Opening	C. Becker/N. Villegas			
9:10	Keynote	TBA	Paul Ralph	Instrumentation and Construct Validity in Software Engineering	To produce more credible research, software engineering research needs more effective measurement. Software engineering studies overwhelming rely on unvalidated and oversimplified metrics and scales for often complicated, multidimensional constructs. Latent variables are often presented as directly measurable and estimated using a single, dubious metric (e.g. NCLOC, fault density) or question (e.g., "How satisfied are you with your productivity?"). Terms like "construct validity" and "Likert scale" appear widely misunderstood. Fortunately, instrument development is an intense area of research in reference disciplines (especially psychology), and there exist fairly straightforward techniques for creating and validating metrics and scales. In this keynote address, Dr. Ralph will unpack the meaning of construct validity, its underlying philosophy, and illustrate practical techniques for validating metrics and scales.
10:00	Panel - Industry expert	Jennifer Schachter			
10:30	Coffe break				
11:00	Intelligent Software Systems	TBA	Song Wang (New Faculty Talk. York U.)	Machine Learning Powered Software Bug Hunting	This talk will explores the feasibility of powering up software reliability with machine learning techniques and presents a suite of machine learning based techniques to improve existing software reliability practices (i.e., bug prediction, bug detection, testing, and code review) for helping developers find software bugs.
11:40			Heng Li (Queen's U)	Towards Intelligent Operations of Large-Scale Software Systems	Modern large-scale software systems are growing rapidly in size and complexity. In the meanwhile, operations of large-scale systems are generating more and more monitoring data, such as metrics, events, and alerts. It becomes increasingly challenging for practitioners to collect, manage, analyze, and leverage such big operational data. To help practitioners overcome such challenges, we proposed intelligent approaches that automatically assist practitioners in their operations of large-scale systems (i.e., intelligent operations). Our approaches aim to achieve intelligent operations along two directions: 1) improving the monitoring code of large-scale systems, and 2) leveraging the monitoring data of large-scale systems. For example, along the first direction, we leveraged semantic topics of the source code to provide automated suggestions about “where to log”. Along the second direction, we leveraged the monitoring data to automatically configure the parameters of large-scale software systems, which significantly reduced human costs in the configuring of the parameters.
11:55			Raul Ivan Perez Martell (UVic)	Shortcomings in the current state of promoter classification using frameworks from machine learning	Understanding DNA sequences has been an ongoing struggle within bioinformatics research. Being able to classify it by function has not yet been possible without biological experiments. We focus on gene regulation, more specifically promoters regions. The data is used to train neural networks to classify 300-length sequences as either promoters or non-promoters. Previous works have focused on already curated datasets to both train and evaluate their models using cross-validation. Their works show how these previous state-of-the-art models have high performing metrics on their respective cross-validation datasets. We argue how those trained models don’t translate well to realistic scenarios by comparing them against each other. We tested each model on different datasets from the literature and on a more realistic dataset containing sequenced human and mouse chromosomes references from the UCSC genome browser database, showing the limitations of these promoter classification approaches. This talk is based on my Master's Thesis research to be defended soon.
12:15	Lunch				*(potentially, an announcement could be scheduled at the end of the break, before the keynote begins)*
13:15	Software Processes and Human Aspects	TBA	Fabian Fagerholm (New Faculty Talk, UoT/Aalto University)	Studying Software Engineering as a Human-Centred Social Activity: Emotions, Values, and Decision-making	There is newfound interest in Software Engineering research on investigating social and behavioural processes that govern individual and group work in the field. Many studies focus on human factors and aspects as predictors or influencers on productivity or quality. Simultaneously, there is growing interest towards investigating fundamental questions around software development as a human and social activity. Such research broadens the scope of SE research from immediate practical concerns for today's organisations to questions regarding, e.g., the experience of being a software developer and the broader implications of software systems design for society in the longer term. This requires researchers to think broadly in terms of research methodology and participation, and often requires a multidisciplinary mindset. In this talk, I will draw from my research collaborations on team performance, emotions, values, and decision-making among software developers to explore perspectives on software development as a human-centred, social activity.

Consortium for Software Engineering Research
2019 Fall Meeting

November 3rd, 2019 - Markham, ON

Time	Session	Chair	Speaker	Title	Abstract
13:55			Curtis McCord (UoT)	Critical Requirements Engineering in Practice	The design of software systems inevitably enacts normative boundaries around the site of intervention. These boundaries are, in part, a reflection of the values, ethics, power, and politics of the situation and the process of design itself. This paper argues that Requirements Engineering (RE) require more robust frameworks and techniques to navigate the values implicit in systems design work. To this end, we present the findings from a case of action research where we employed Critical Systems Heuristics (CSH), a framework from Critical Systems Thinking (CST) during requirements gathering for Homesound, a system to safeguard elderly people living alone while protecting their autonomy. We use categories from CSH to inform expert interviews and reflection, showing how CSH can be simply combined with RE techniques (such as the Volere template) to explore and reveal the value-judgements underlying requirements.
14:10			Sedef Akinli Kocak (Vector Institute)	Raising awareness of potential Sustainability Effects of Software Systems in Requirements Engineering	Integrating novel software systems in our society, economy, and environment can have far-reaching effects. As a result, software systems should be designed in such a way as to maintain or improve the sustainability of the socio-technical system of their destination. However, a paradigm shift is required to raise awareness of software professionals on the potential sustainability effects of software systems. While Requirements Engineering is considered the key to driving this change, requirements engineers lack the knowledge, experience and methodological support for doing so. This talk presents a question-based framework for raising awareness of the potential effects of software systems on sustainability, as the first step towards enabling the required paradigm shift. A feasibility study of the framework was carried out with two groups of computer science students. The results of the study indicate that the framework helps enable discussions about potential effects that software systems could have on sustainability.
14:25			Naser Ezzati Jivan (Polytechnique M)	Observability-Driven Software Development Process	Software observability is a concept that is essential to understand how software is behaving in production. Basically, software observability deals with the ability to observe the possible internal state changes and software modules reaction to the different inputs that are fed during the software execution. By having internal state changes and interactions as the result of software observability, the runtime software defects can be monitored, detected and reasoned. Software observability can be reached through different techniques including being able to instrument, trace and log the software execution. Although some people define observability as a non-functional requirement, we discuss the necessity and importance of aligning the software development process to incorporate the observability as an important add-on to its different phases. We highlight the certain steps that should be included in software specification, design, implementation, testing and deployment to attain and enhance the observability of a software product.
14:40			Rabe Abdalkareem (Queen's U)	A Machine Learning Approach to Predicate the Semantic Versioning	Semantic versioning is widely used to indicate the level of changes in a release. Unfortunately, there are many cases where developers do not respect the semantic versioning policy, leading to the breakage of dependent applications. To reduce such cases, we propose the use of machine learning techniques to effectively predict the type of a release, i.e., patch, minor, major in order to properly determine the semantic versioning. To perform our prediction, we mine and use a number of factors about a release, such as the complexity and development activities. We use three ML classifiers. To evaluate the performance of the proposed ML approach, we conduct an empirical study on 100 JavaScript packages containing a total of approximately 7,000 releases. We found that, on average, our approach achieves accuracy values between 0.67 and 0.79. We find that factors related to the complexity of change in a release are the best predictors.
14:55			Timothy C. Lethbridge (UofO)	Using Umple as an Educational and Research Platform in Software Engineering	Although Umple is at its core a modeling tool, generating code from UML class and state models, it also incorporates a rich suite of other features leveraging its dual textual/diagram nature. We will show how mixsets, aspects, filters and traits work together synergistically to allow management of models, code, testcases and diagrams for large systems and product lines. Umple is designed to make learning and using modeling easy: It is targeted at open-source developers and students. It is written in itself, and its 60 student developers have all followed a test-driven, model-driven, agile approach. As such can be used as a case study for a wide variety of best practices in software engineering. Umple is celebrating its 12th year; its online version serves over 200,000 user sessions annually. We presented Umple in 2010 and 2013 at CSER; we will focus on the advances since that time.
15:10	Coffe break				*(potentially, an announcement could be scheduled at the end of the break, before the keynote begins)*
15:40	Quality attributes, testing and logging	TBA	Karim Ali (New Faculty Talk, UoA)	Scalable and Precise Detection of Security Vulnerabilities	Precise detection of security vulnerabilities requires static analyses that are context-sensitive, field-sensitive, and flow-sensitive. Context-sensitivity and field-sensitivity can both be expressed as context-free language (CFL) reachability problems. However, solving both CFL problems along the same dataflow path is undecidable. Therefore, most flow-sensitive dataflow analyses over-approximate field-sensitivity by limiting the depth of successive field accesses (i.e., access path) to a certain threshold. Unfortunately, such representation does not scale very well when used to analyze complex programs, and usually produces many false positives in practice. In this talk, I will introduce the novel concept of synchronized pushdown systems (SPDS) that encodes procedure calls/returns and field stores/loads as separate but synchronized CFL reachability problems. I will show how this representation allows SPDS to scale to large codebases, which allowed us to discover real-world security vulnerabilities.

Consortium for Software Engineering Research
2019 Fall Meeting

November 3rd, 2019 - Markham, ON

Time	Session	Chair	Speaker	Title	Abstract
16:20			Mohammadreza Rasolroveicy (Polytechnique M)	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 (Hyperledger Fabric, HyperIdger Burrow and BigchainDB) 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.
16:35			Diego Costa (Concordia U)	How type-safe are open-source Go applications?	Golang (short for Go programming language) is an open-source, compiled programming language developed by Google. The language is praised for its clean syntax, and high-performance making Golang an ideal choice for back-end development. While Golang is type-safe by design, the language ships with a package called Unsafe that offers developers a way around the type-safety of Go programs. The package gives higher flexibility to developers, enabling direct memory manipulation, often used to implement low-level routines. This flexibility, however, comes at a higher risk of runtime errors, chances of non-portability and the loss of compatibility guarantees in the future versions of Go. In this talk, I will present a preliminary analysis of a large-scale study on the usage of the Unsafe package by real-life, open-source Go applications.
16:50			Jinfu Chen (Concordia U.)	An Experience Report of Generating Load Tests Using Log-recovered Workloads at Varying Granularities of User Behaviour	Designing field-representative load tests is an essential step for the quality assurance of large-scale systems. Practitioners may capture user behaviour at different levels of granularity. A coarse-grained load test may miss detailed user behaviour, leading to a non-representative load test; while an extremely fine-grained load test would simply replay user actions step by step, leading to load tests that are costly to develop, execute and maintain. Workload recovery is at core of these load tests. Prior research often captures the workload as the frequency of user actions. However, there exists much valuable information in the context and sequences of user actions. Such richer information would ensure that the load tests that leverage such workloads are more field-representative. In this experience paper, we study the use of different granularities of user behaviour, i.e., basic user actions, basic user actions with contextual information and user action sequences with contextual information, when recovering workloads for use in the load testing of large-scale systems. We propose three approaches that are based on the three granularities of user behaviour and evaluate our approaches on four subject systems, namely Apache James, OpenMRS, Google Borg, and an ultra-large-scale industrial system (SA) from Alibaba. Our results show that our approach that is based on user action sequences with contextual information outperforms the other two approaches and can generate more representative load tests with similar throughput and CPU usage to the original field workload (i.e., mostly statistically insignificant or with small/trivial effect sizes). Such representative load tests are generated only based on a small number of clusters of users, leading to a low cost of conducting/maintaining such tests. Finally, we demonstrate that our approaches can detect injected users in the original field workloads with high precision and recall. Our paper demonstrates the importance of user action sequences with contextual information in the workload recovery of large-scale systems.
17:05			Boyuan Chen (York U.)	Extracting and studying the Logging-Code-Issue-Introducing changes in Java-based large-scale open source software systems	Execution logs, which are generated by logging code, are widely used in modern software projects for tasks like monitoring, debugging, and remote issue resolution. Ineffective logging would cause confusion, lack of information during problem diagnosis, or even system crash. However, it is challenging to develop and maintain logging code, as it inter-mixes with the feature code. Furthermore, unlike feature code, it is very challenging to verify the correctness of logging code. Currently developers usually rely on their intuition when performing their logging activities. There are no well established logging guidelines in research and practice. In this paper, we intend to derive such guidelines through mining the historical logging code changes. In particular, we have extracted and studied the Logging-Code-Issue-Introducing (LCII) changes in six popular large-scale Java-based open source software systems. Preliminary studies on this dataset show that: (1) both co-changed and independently changed logging code changes can contain fixes to the LCII changes; (2) the complexity of fixes to LCII changes are similar to regular logging code updates; (3) it takes longer for developers to fix logging code issues than regular bugs; and (4) the state-of-the-art logging code issue detection tools can only detect a small fraction (3%) of the LCII changes. This highlights the urgent need for this area of research and the importance of such a dataset.
17:20	1 minute poster presentation - Closing and announcements				
17:40	Reception/posters				
19:00	End of the event				